Kobe Bryant: Shot Clustering

Business Understanding

The database selected is a compilation of individual shot metrics and statistics from Kobe Bryant's entire career. Starting with data points from his rookie season (1996/1997) to his final season (2015/2016). The main purpose of the aggregated database is a classification model of a shot to be made or missed, but also for working classification basics, feature engineering and time-series analysis.

Accuracy will be the best way to measure the effectiveness of the algorithm we are developing. The ultimate goal is to use clustering as a variable reduction mechanism to predict the outcome of a shot with an accuracy level better than random chance. In order to determine the best model we need to have the highest level of accuracy using the Kobe Bryant shot data.

https://msdn.microsoft.com/en-us/library/ms174493.aspx (https://msdn.microsoft.com/en-us/library/ms174493.aspx)

Data Understanding

The dataset contains 20 seasons of Kobe Bryant's shot data from the regular season and playoff games (30,697 shots). 5,000 of these shots have not been marked as to whether or not the shot was made (shot_made_flag), these records will be placed into a testing data set. Our initial dataset had 25 variables with many of the variables over-lapping or unsuitable for statistical analysis. By altering feature measures, adding variables and a few other twists were able to condense our data set to 18 variables.

Our 18 variables include:

loc_x: X-coordinate regarding the location of the shot on the court (sideline to sideline)

loc_y: Y coordinate regarding the location of the shot on the court (baseline to midline)

period: The quarter of the game

playoffs: Whether or not the team was in the playoffs (0=no, 1=yes)

shot distance: Distance from the basket the shoot took place in feet

shot_made_flag: Whether or not the shot was made (0=no, 1=yes)

game_date: the date on which the game occurred

shot_id: An unique identifier of each shot (mainly to be used for further investigation)

time_remaining_sec: The remaining time, in seconds, in a quarter (period)

home_game: Whether the game played was classified as a home game (0=no, 1=yes)

two_pt_fg: Whether the shot take was classified as a 2-point field goal (0=no, 1=yes)

season_id: The corresponding number seasons Kobe had finished in the NBA

action_type_id: the type of action (during) and the type of shot being taken (i.e. running

jump shot, reverse dunk shot, turnaround jump shot)

shot_zone_id: unique identifier of a zone on the court (i.e. Mid-Range, Restricted Area)

shot_area_id: unique identifier of an area on the court (i.e. Left, Right, Center)

opponent_id: Team identification number (i.e. Seattle, Charlotte, San Antonio)

shot_range_id: range identifier from the basket, based on intervals of 8 feet (i.e. 16-24 ft.)

game_event_id: the event number within a game that a shot took place.

In [1]:
```
# Load the Kobe dataset
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import random
from sklearn.cross_validation import StratifiedKFold, cross_val_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.cluster import KMeans
%matplotlib inline

df = pd.read_csv('data.csv') # read in the csv file

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 30697 entries, 0 to 30696
Data columns (total 25 columns):
action_type           30697 non-null object
combined_shot_type    30697 non-null object
game_event_id         30697 non-null int64
game_id               30697 non-null int64
lat                   30697 non-null float64
loc_x                 30697 non-null int64
loc_y                 30697 non-null int64
lon                   30697 non-null float64
minutes_remaining     30697 non-null int64
period                30697 non-null int64
playoffs              30697 non-null int64
season                30697 non-null object
seconds_remaining     30697 non-null int64
shot_distance         30697 non-null int64
shot_made_flag        25697 non-null float64
shot_type             30697 non-null object
shot_zone_area        30697 non-null object
shot_zone_basic       30697 non-null object
shot_zone_range       30697 non-null object
team_id               30697 non-null int64
team_name             30697 non-null object
game_date             30697 non-null object
matchup               30697 non-null object
opponent              30697 non-null object
shot_id               30697 non-null int64
dtypes: float64(3), int64(11), object(11)
memory usage: 6.1+ MB
```

In [2]:
```
# printing data to better understand what we are working with

df.head(10)
```

Out[2]:

	action_type	combined_shot_type	game_event_id	game_id	lat	loc_x
0	Jump Shot	Jump Shot	10	20000012	33.9723	167
1	Jump Shot	Jump Shot	12	20000012	34.0443	-157
2	Jump Shot	Jump Shot	35	20000012	33.9093	-101
3	Jump Shot	Jump Shot	43	20000012	33.8693	138
4	Driving Dunk Shot	Dunk	155	20000012	34.0443	0
5	Jump Shot	Jump Shot	244	20000012	34.0553	-145
6	Layup Shot	Layup	251	20000012	34.0443	0
7	Jump Shot	Jump Shot	254	20000012	34.0163	1
8	Jump Shot	Jump Shot	265	20000012	33.9363	-65
9	Running Jump Shot	Jump Shot	294	20000012	33.9193	-33

10 rows × 25 columns

In [3]:
```
# Data Cleansing

# Modifying season to make it numeric
# Sorting our data frame based on Season, Date and Shot
df['season'] = df.season.str.split('-').str.get(0)
df['season'] = df['season'].astype(np.int)
df.sort_values(['season', 'game_date', 'shot_id'], ascending=[True, Tru
e, True], inplace=True)

df['time_remaining_sec'] = (df['minutes_remaining']*60) + df['seconds_re
maining']
df['home_game'] = (df['matchup'].str.contains('vs.')).astype(int)
df['two_pt_fg'] = (df['shot_type'].str.contains('2PT')).astype(int)

# Giving IDs to categorical variables
season_map = {action: i for i, action in enumerate(df.season.unique())}
df['season_id'] = df.season.map(season_map)
print('Number of Seasons: %d' % (len(season_map)))

shot_type_map = {action: i for i, action in enumerate(df.combined_shot_t
ype.unique())}
df['shot_type_id'] = df.combined_shot_type.map(shot_type_map)
print('Number of Shot Types: %d' % (len(shot_type_map)))

shot_area_map = {action: i for i, action in enumerate(df.shot_zone_area.
unique())}
df['shot_area_id'] = df.shot_zone_area.map(shot_area_map)
print('Number of Shot Areas: %d' % (len(shot_area_map)))

action_map = {action: i for i, action in enumerate(df.action_type.unique
())}
df['action_type_id'] = df.action_type.map(action_map)
print('Number of Action Types: %d' % (len(action_map)))

shot_zone_map = {action: i for i, action in enumerate(df.shot_zone_basi
c.unique())}
df['shot_zone_id'] = df.shot_zone_basic.map(shot_zone_map)
print('Number of Shot Zones: %d' % (len(shot_zone_map)))

opponent_map = {action: i for i, action in enumerate(df.opponent.unique
())}
df['opponent_id'] = df.opponent.map(opponent_map)
print('Number of Opponents: %d' % (len(opponent_map)))

shot_range_map = {action: i for i, action in enumerate(df.shot_zone_rang
e.unique())}
df['shot_range_id'] = df.shot_zone_range.map(shot_range_map)
print('Number of Shot Ranges: %d' % (len(shot_range_map)))
```

```
Number of Seasons: 20
Number of Shot Types: 6
Number of Shot Areas: 6
Number of Action Types: 57
Number of Shot Zones: 7
Number of Opponents: 33
Number of Shot Ranges: 5
```

In [4]:
```
# Deleting rows that we will not use

del df['lat']
del df['lon']
del df['game_id']
del df['minutes_remaining']
del df['seconds_remaining']
del df['team_id']
del df['team_name']
del df['shot_type']
del df['matchup']
del df['season']
del df['combined_shot_type']
del df['shot_zone_area']
del df['action_type']
del df['opponent']
del df['shot_zone_basic']
del df['shot_zone_range']

df.head(10)
```

Out[4]:

	game_event_id	loc_x	loc_y	period	playoffs	shot_distance	shot_made
22901	102	-140	116	1	0	18	0
22902	127	-131	97	2	0	16	0
22903	124	-142	181	2	0	23	1
22904	144	0	0	2	0	0	0
22905	151	-10	138	2	0	13	1
22906	157	75	177	2	0	19	NaN
22907	226	-64	223	2	0	23	1
22908	321	0	0	3	0	0	NaN
22909	334	-79	177	3	0	19	0
22910	337	-103	207	3	0	23	1

In [5]:
```
# Creating 2 new Dataframes

kobe_reg = df.loc[-df['shot_made_flag'].isin([0, 1])]
kobe_shots= df.loc[df['shot_made_flag'].isin([0, 1])]
```

In [6]:
```
# Create dataframes for Diffent Seasons

# Rookie Season
kobe_df0_pre = kobe_reg.loc[kobe_reg['season_id'].isin([0])]
print kobe_df0_pre.shape

# Adjusting to the league
kobe_df0 = kobe_shots.loc[kobe_shots['season_id'].isin([0])]
kobe_df1_pre = kobe_reg.loc[kobe_reg['season_id'].isin([1, 2])]
print kobe_df1_pre.shape

# Coach/Philosopy Transition
kobe_df1 = kobe_shots.loc[kobe_shots['season_id'].isin([2])]
kobe_df2_pre = kobe_reg.loc[kobe_reg['season_id'].isin([3])]
print kobe_df2_pre.shape

# Start of Phil Jackson era
kobe_df2 = kobe_shots.loc[kobe_shots['season_id'].isin([3])]
kobe_df3_pre = kobe_reg.loc[kobe_reg['season_id'].isin([4, 5])]
print kobe_df3_pre.shape

# Triangle era
kobe_df3 = kobe_shots.loc[kobe_shots['season_id'].isin([3, 4, 5])]
kobe_df4_pre = kobe_reg.loc[kobe_reg['season_id'].isin([6, 7, 8, 9, 10,
11, 12, 13, 14, 15])]
print kobe_df4_pre.shape

# Mike D'Antoni era
kobe_df4 = kobe_shots.loc[kobe_shots['season_id'].isin([15])]
kobe_df5_pre = kobe_reg.loc[kobe_reg['season_id'].isin([16, 17])]
print kobe_df5_pre.shape

# Injury/Coach Transition
kobe_df5 = kobe_shots.loc[kobe_shots['season_id'].isin([16, 17])]
kobe_df6_pre = kobe_reg.loc[kobe_reg['season_id'].isin([18, 19])]
print kobe_df6_pre.shape
```

```
(94, 19)
(335, 19)
(265, 19)
(613, 19)
(3113, 19)
(279, 19)
(301, 19)
```

Visualization of Attributes – Provide Chart/Graphs Interpretation

Loc_x vs. loc_y (Shot charts – maybe color coordinate for makes)

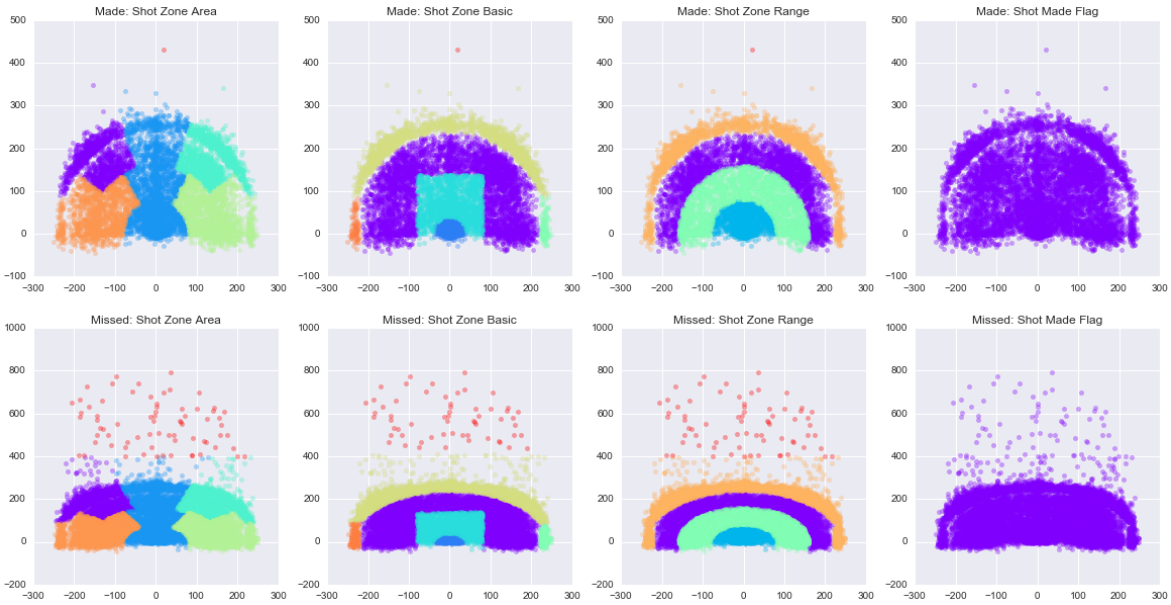
Shots made vs. Event ID

In [7]:
```
def scatter_plot_by_category(data, feat):
    alpha = 0.35
    gs = data.groupby(feat)
    cs = plt.cm.rainbow(np.linspace(0, 1, len(gs)))
    for g, c in zip(gs, cs):
        plt.scatter(g[1].loc_x, g[1].loc_y, color=c, alpha=alpha)
```

All Data

In [8]:
```
plt.figure(figsize=(20,10))

h1 = df.loc[df.shot_made_flag == 1]
h2 = df.loc[df.shot_made_flag == 0]

# shot_zone_area
plt.subplot(241)
scatter_plot_by_category(h1, 'shot_area_id')
plt.title('Made: Shot Zone Area')

# shot_zone_basic
plt.subplot(242)
scatter_plot_by_category(h1, 'shot_zone_id')
plt.title('Made: Shot Zone Basic')

# shot_zone_range
plt.subplot(243)
scatter_plot_by_category(h1, 'shot_range_id')
plt.title('Made: Shot Zone Range')

# Made / Miss
plt.subplot(244)
scatter_plot_by_category(h1, 'shot_made_flag')
plt.title('Made: Shot Made Flag')

# shot_zone_area
plt.subplot(245)
scatter_plot_by_category(h2, 'shot_area_id')
plt.title('Missed: Shot Zone Area')

# shot_zone_basic
plt.subplot(246)
scatter_plot_by_category(h2, 'shot_zone_id')
plt.title('Missed: Shot Zone Basic')

# shot_zone_range
plt.subplot(247)
scatter_plot_by_category(h2, 'shot_range_id')
plt.title('Missed: Shot Zone Range')

# Made / Miss
plt.subplot(248)
scatter_plot_by_category(h2, 'shot_made_flag')
plt.title('Missed: Shot Made Flag')
```

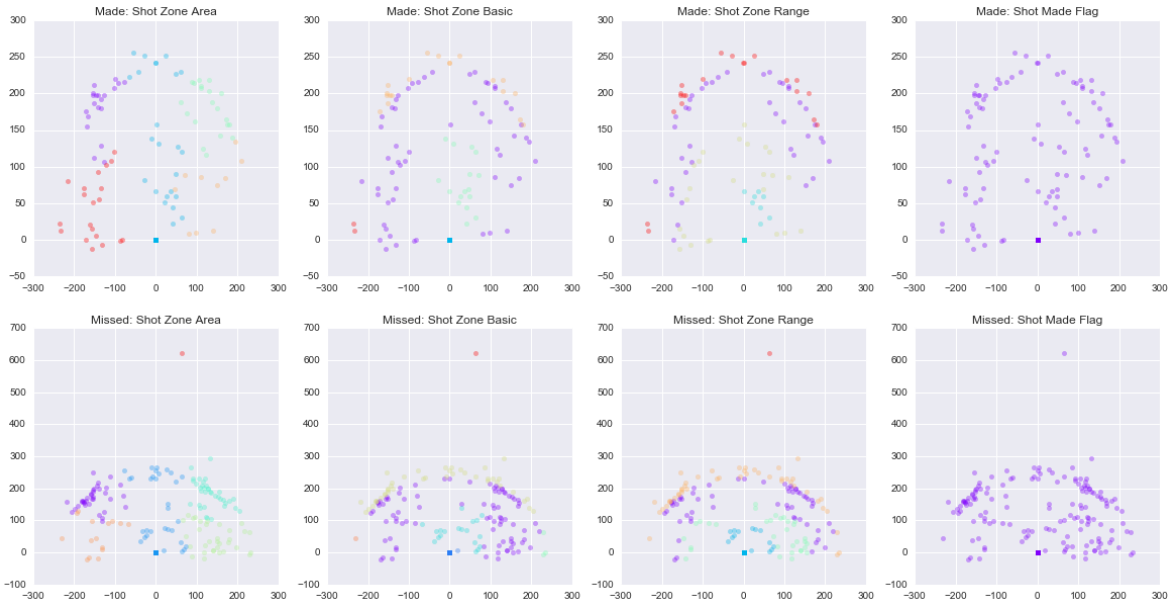
Out[8]: <matplotlib.text.Text at 0x1136bd650>



Model 0: Rookie Season

In [9]:
```
plt.figure(figsize=(20,10))

h1 = kobe_df0.loc[kobe_df0.shot_made_flag == 1]
h2 = kobe_df0.loc[kobe_df0.shot_made_flag == 0]

# shot_zone_area
plt.subplot(241)
scatter_plot_by_category(h1, 'shot_area_id')
plt.title('Made: Shot Zone Area')

# shot_zone_basic
plt.subplot(242)
scatter_plot_by_category(h1, 'shot_zone_id')
plt.title('Made: Shot Zone Basic')

# shot_zone_range
plt.subplot(243)
scatter_plot_by_category(h1, 'shot_range_id')
plt.title('Made: Shot Zone Range')

# Made / Miss
plt.subplot(244)
scatter_plot_by_category(h1, 'shot_made_flag')
plt.title('Made: Shot Made Flag')

# shot_zone_area
plt.subplot(245)
scatter_plot_by_category(h2, 'shot_area_id')
plt.title('Missed: Shot Zone Area')

# shot_zone_basic
plt.subplot(246)
scatter_plot_by_category(h2, 'shot_zone_id')
plt.title('Missed: Shot Zone Basic')

# shot_zone_range
plt.subplot(247)
scatter_plot_by_category(h2, 'shot_range_id')
plt.title('Missed: Shot Zone Range')

# Made / Miss
plt.subplot(248)
scatter_plot_by_category(h2, 'shot_made_flag')
plt.title('Missed: Shot Made Flag')
```

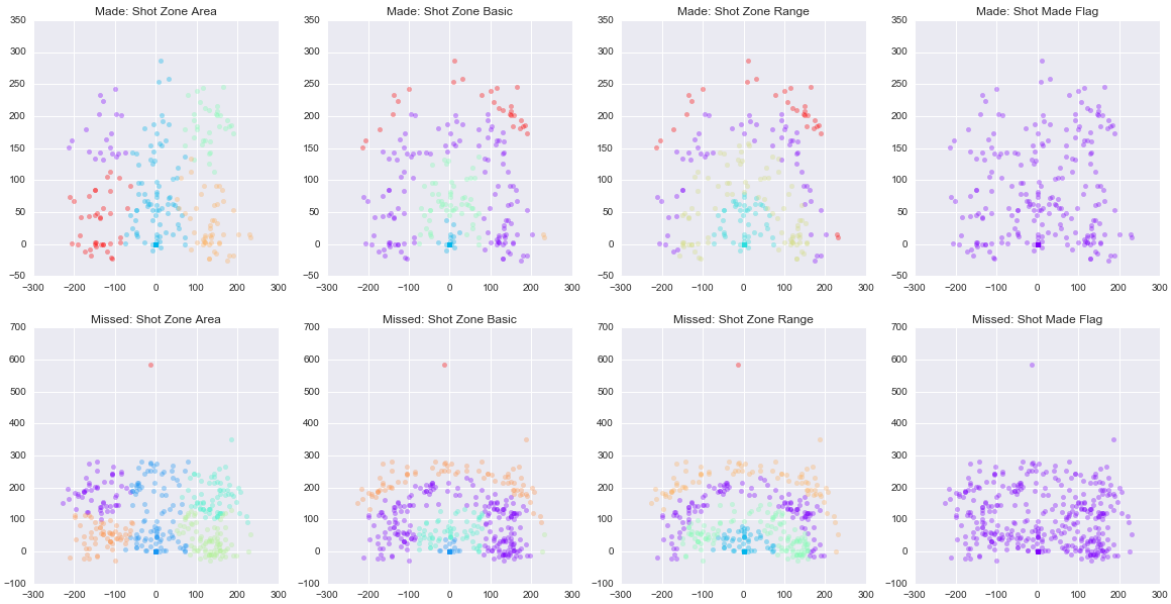
```
Out[9]: <matplotlib.text.Text at 0x113e66e90>
```



Model 1: Adjusting To The League

In [10]:
```
plt.figure(figsize=(20,10))

h1 = kobe_df1.loc[kobe_df1.shot_made_flag == 1]
h2 = kobe_df1.loc[kobe_df1.shot_made_flag == 0]

# shot_zone_area
plt.subplot(241)
scatter_plot_by_category(h1, 'shot_area_id')
plt.title('Made: Shot Zone Area')

# shot_zone_basic
plt.subplot(242)
scatter_plot_by_category(h1, 'shot_zone_id')
plt.title('Made: Shot Zone Basic')

# shot_zone_range
plt.subplot(243)
scatter_plot_by_category(h1, 'shot_range_id')
plt.title('Made: Shot Zone Range')

# Made / Miss
plt.subplot(244)
scatter_plot_by_category(h1, 'shot_made_flag')
plt.title('Made: Shot Made Flag')

# shot_zone_area
plt.subplot(245)
scatter_plot_by_category(h2, 'shot_area_id')
plt.title('Missed: Shot Zone Area')

# shot_zone_basic
plt.subplot(246)
scatter_plot_by_category(h2, 'shot_zone_id')
plt.title('Missed: Shot Zone Basic')

# shot_zone_range
plt.subplot(247)
scatter_plot_by_category(h2, 'shot_range_id')
plt.title('Missed: Shot Zone Range')

# Made / Miss
plt.subplot(248)
scatter_plot_by_category(h2, 'shot_made_flag')
plt.title('Missed: Shot Made Flag')
```

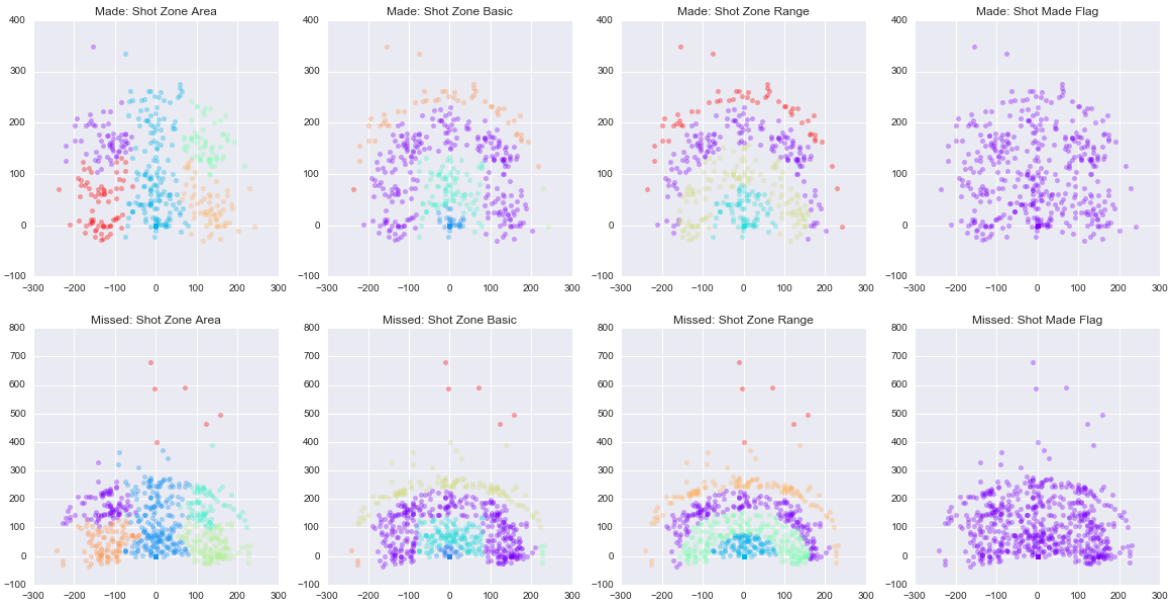
Out[10]: <matplotlib.text.Text at 0x115ab2450>



Model 2: Coach/Philosopy Transition

In [11]:
```
plt.figure(figsize=(20,10))

h1 = kobe_df2.loc[kobe_df2.shot_made_flag == 1]
h2 = kobe_df2.loc[kobe_df2.shot_made_flag == 0]

# shot_zone_area
plt.subplot(241)
scatter_plot_by_category(h1, 'shot_area_id')
plt.title('Made: Shot Zone Area')

# shot_zone_basic
plt.subplot(242)
scatter_plot_by_category(h1, 'shot_zone_id')
plt.title('Made: Shot Zone Basic')

# shot_zone_range
plt.subplot(243)
scatter_plot_by_category(h1, 'shot_range_id')
plt.title('Made: Shot Zone Range')

# Made / Miss
plt.subplot(244)
scatter_plot_by_category(h1, 'shot_made_flag')
plt.title('Made: Shot Made Flag')

# shot_zone_area
plt.subplot(245)
scatter_plot_by_category(h2, 'shot_area_id')
plt.title('Missed: Shot Zone Area')

# shot_zone_basic
plt.subplot(246)
scatter_plot_by_category(h2, 'shot_zone_id')
plt.title('Missed: Shot Zone Basic')

# shot_zone_range
plt.subplot(247)
scatter_plot_by_category(h2, 'shot_range_id')
plt.title('Missed: Shot Zone Range')

# Made / Miss
plt.subplot(248)
scatter_plot_by_category(h2, 'shot_made_flag')
plt.title('Missed: Shot Made Flag')
```

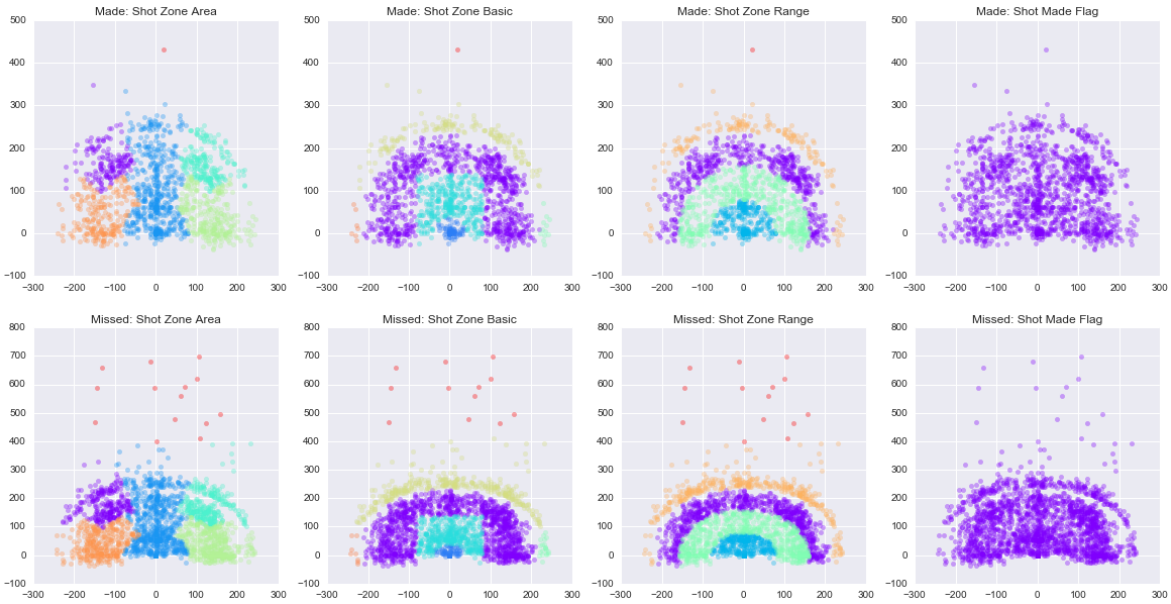
```
Out[11]: <matplotlib.text.Text at 0x116571990>
```



Model 3: Start Of Phil Jackson Era

In [12]:
```
plt.figure(figsize=(20,10))

h1 = kobe_df3.loc[kobe_df3.shot_made_flag == 1]
h2 = kobe_df3.loc[kobe_df3.shot_made_flag == 0]

# shot_zone_area
plt.subplot(241)
scatter_plot_by_category(h1, 'shot_area_id')
plt.title('Made: Shot Zone Area')

# shot_zone_basic
plt.subplot(242)
scatter_plot_by_category(h1, 'shot_zone_id')
plt.title('Made: Shot Zone Basic')

# shot_zone_range
plt.subplot(243)
scatter_plot_by_category(h1, 'shot_range_id')
plt.title('Made: Shot Zone Range')

# Made / Miss
plt.subplot(244)
scatter_plot_by_category(h1, 'shot_made_flag')
plt.title('Made: Shot Made Flag')

# shot_zone_area
plt.subplot(245)
scatter_plot_by_category(h2, 'shot_area_id')
plt.title('Missed: Shot Zone Area')

# shot_zone_basic
plt.subplot(246)
scatter_plot_by_category(h2, 'shot_zone_id')
plt.title('Missed: Shot Zone Basic')

# shot_zone_range
plt.subplot(247)
scatter_plot_by_category(h2, 'shot_range_id')
plt.title('Missed: Shot Zone Range')

# Made / Miss
plt.subplot(248)
scatter_plot_by_category(h2, 'shot_made_flag')
plt.title('Missed: Shot Made Flag')
```

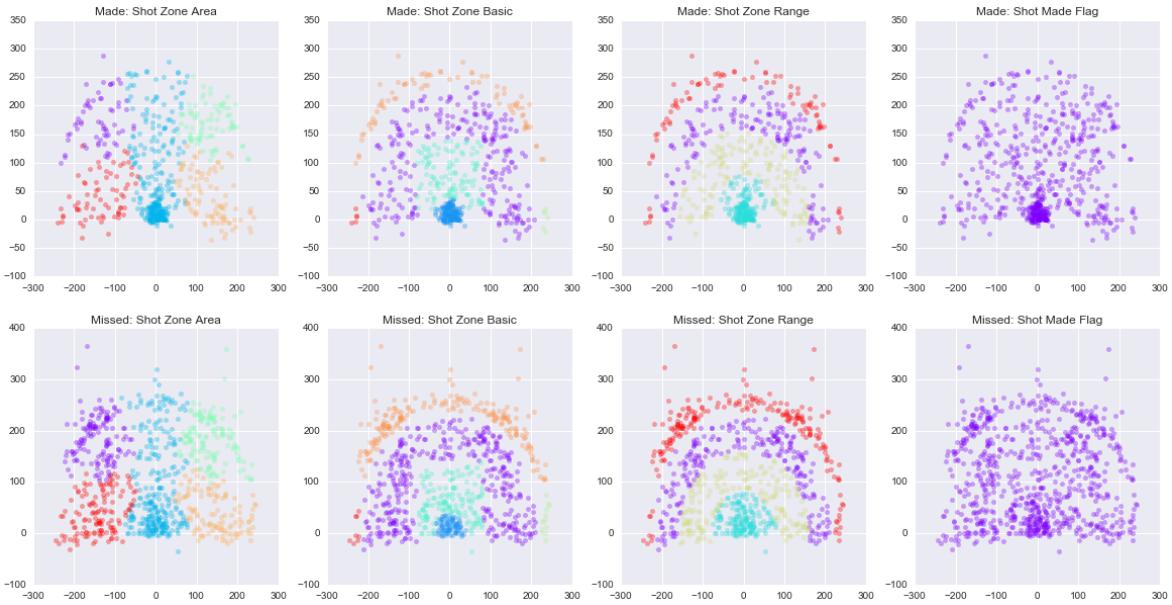
Out[12]: <matplotlib.text.Text at 0x1170c6990>



Model 4: Triangle Era

In [13]:
```
plt.figure(figsize=(20,10))

h1 = kobe_df4.loc[kobe_df4.shot_made_flag == 1]
h2 = kobe_df4.loc[kobe_df4.shot_made_flag == 0]

# shot_zone_area
plt.subplot(241)
scatter_plot_by_category(h1, 'shot_area_id')
plt.title('Made: Shot Zone Area')

# shot_zone_basic
plt.subplot(242)
scatter_plot_by_category(h1, 'shot_zone_id')
plt.title('Made: Shot Zone Basic')

# shot_zone_range
plt.subplot(243)
scatter_plot_by_category(h1, 'shot_range_id')
plt.title('Made: Shot Zone Range')

# Made / Miss
plt.subplot(244)
scatter_plot_by_category(h1, 'shot_made_flag')
plt.title('Made: Shot Made Flag')

# shot_zone_area
plt.subplot(245)
scatter_plot_by_category(h2, 'shot_area_id')
plt.title('Missed: Shot Zone Area')

# shot_zone_basic
plt.subplot(246)
scatter_plot_by_category(h2, 'shot_zone_id')
plt.title('Missed: Shot Zone Basic')

# shot_zone_range
plt.subplot(247)
scatter_plot_by_category(h2, 'shot_range_id')
plt.title('Missed: Shot Zone Range')

# Made / Miss
plt.subplot(248)
scatter_plot_by_category(h2, 'shot_made_flag')
plt.title('Missed: Shot Made Flag')
```

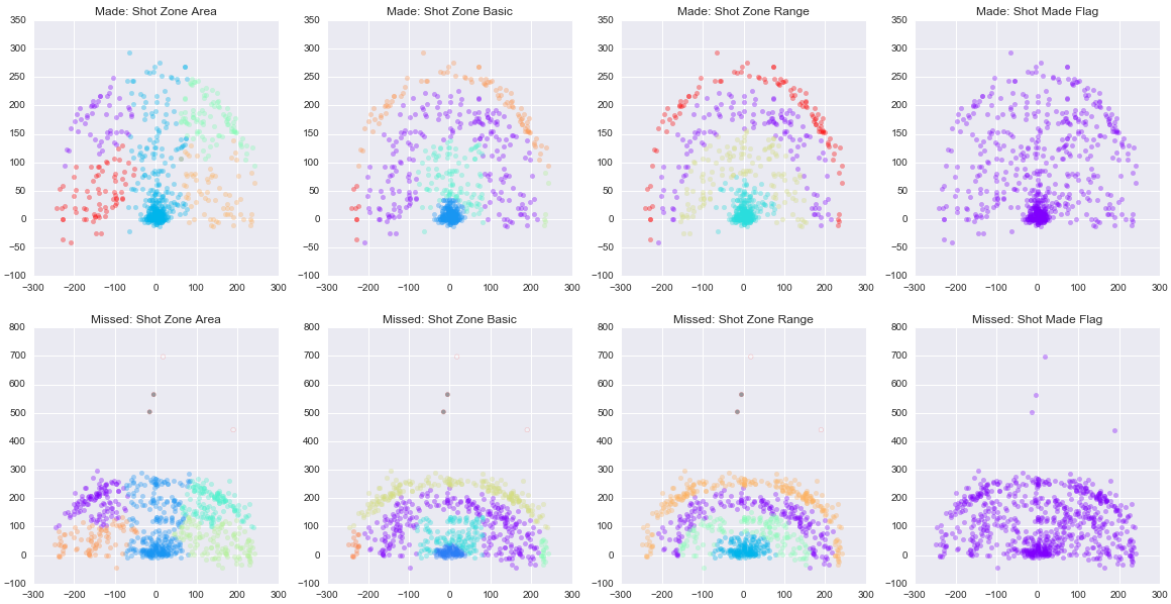
```
Out[13]: <matplotlib.text.Text at 0x117c08fd0>
```



Model 5: Mike D'Antoni Era

In [14]:
```
plt.figure(figsize=(20,10))

h1 = kobe_df5.loc[kobe_df5.shot_made_flag == 1]
h2 = kobe_df5.loc[kobe_df5.shot_made_flag == 0]

# shot_zone_area
plt.subplot(241)
scatter_plot_by_category(h1, 'shot_area_id')
plt.title('Made: Shot Zone Area')

# shot_zone_basic
plt.subplot(242)
scatter_plot_by_category(h1, 'shot_zone_id')
plt.title('Made: Shot Zone Basic')

# shot_zone_range
plt.subplot(243)
scatter_plot_by_category(h1, 'shot_range_id')
plt.title('Made: Shot Zone Range')

# Made / Miss
plt.subplot(244)
scatter_plot_by_category(h1, 'shot_made_flag')
plt.title('Made: Shot Made Flag')

# shot_zone_area
plt.subplot(245)
scatter_plot_by_category(h2, 'shot_area_id')
plt.title('Missed: Shot Zone Area')

# shot_zone_basic
plt.subplot(246)
scatter_plot_by_category(h2, 'shot_zone_id')
plt.title('Missed: Shot Zone Basic')

# shot_zone_range
plt.subplot(247)
scatter_plot_by_category(h2, 'shot_range_id')
plt.title('Missed: Shot Zone Range')

# Made / Miss
plt.subplot(248)
scatter_plot_by_category(h2, 'shot_made_flag')
plt.title('Missed: Shot Made Flag')
```

Out[14]: <matplotlib.text.Text at 0x1186db8d0>



Building Our Benchmark

Model 0: Rookie Season

In [63]:
```
# Randomly choosing a value for Rookie season missing data

submission_bench = pd.DataFrame({'shot_id': kobe_df0_pre.shot_id})
submission_bench['shot_made_flag'] = np.random.uniform(0.01, 0.99, len(s
ubmission_bench))
submission_bench.head(10)
```

Out[63]:

	shot_id	shot_made_flag
22906	22907	0.467664
22908	22909	0.213607
22925	22926	0.935771
22926	22927	0.610965
22929	22930	0.448001
22933	22934	0.478125
22936	22937	0.259852
22938	22939	0.098096
22948	22949	0.454157
22950	22951	0.135222

Model 1: Adjusting to the league

***Top 5 features will vary based run**

In [18]:
```
# Feature Selection based off of Random Forest

X = kobe_df0[['game_event_id', 'loc_x', 'loc_y', 'period', 'playoffs',
'shot_distance', 'time_remaining_sec', 'home_game', 'two_pt_fg', 'season
_id', 'action_type_id', 'shot_area_id', 'shot_zone_id', 'opponent_id',
'shot_range_id', 'shot_type_id']]
y = kobe_df0['shot_made_flag']
names=X.dtypes.index

clf = RandomForestClassifier(n_jobs=-1, n_estimators=50)

scores = []
for i in range(X.shape[1]):
     score = cross_val_score(clf, X, y, scoring="roc_auc", cv=10)
     scores.append((round(np.mean(score), 3), names[i]))
print (sorted(scores, reverse=True))
```

```
[(0.623, 'shot_area_id'), (0.623, 'season_id'), (0.622, 'loc_x'), (0.61
4, 'shot_zone_id'), (0.614, 'shot_distance'), (0.612, 'shot_type_id'),
(0.61, 'shot_range_id'), (0.608, 'two_pt_fg'), (0.608, 'opponent_id'),
(0.606, 'action_type_id'), (0.605, 'playoffs'), (0.604, 'period'), (0.6
04, 'home_game'), (0.602, 'loc_y'), (0.601, 'time_remaining_sec'), (0.
6, 'game_event_id')]
```

In [64]:
```
# Training model with Top 5 Features from above
clf = RandomForestClassifier(n_jobs=-1, n_estimators=70)

train_y = kobe_df0['shot_made_flag']
train_X = kobe_df0[['shot_area_id', 'season_id', 'loc_x', 'shot_zone_i
d', 'shot_distance']]
cv = StratifiedKFold(train_y, n_folds=10)

acc = cross_val_score(clf, train_X, y=train_y, cv=cv)
print "Average Accuracy On Training Data= ", acc.mean()*100, "+-", acc.s
td()*100

clf.fit(train_X, train_y)

# Predict for Era 1

test_X = kobe_df1_pre[['shot_area_id', 'season_id', 'loc_x', 'shot_zone_
id', 'shot_distance', 'shot_id']]
predictions = clf.predict_proba(test_X.drop('shot_id', 1))
submission_b1 = pd.DataFrame({'shot_id': test_X.shot_id, 'shot_made_fla
g': predictions[:, 1]})

sub = [submission_bench, submission_b1]
submission_bench = pd.concat(sub)

submission_b1.head(10)
```

```
Average Accuracy On Training Data=  50.6943319838 +- 6.91509382995
```

Out[64]:

	shot_id	shot_made_flag
23341	23342	0.785714
23343	23344	0.428571
23349	23350	0.328571
23366	23367	0.495063
23367	23368	0.157143
23373	23374	0.228571
23386	23387	0.385714
23388	23389	0.495063
23398	23399	0.376190
23402	23403	0.083333

Model 2: Coach/Philosopy Transition

***Top 5 features will vary based run**

In [20]:
```
# Feature Selection based off of Random Forest

X = kobe_df1[['game_event_id', 'loc_x', 'loc_y', 'period', 'playoffs',
'shot_distance', 'time_remaining_sec', 'home_game', 'two_pt_fg', 'season
_id', 'action_type_id', 'shot_area_id', 'shot_zone_id', 'opponent_id',
'shot_range_id', 'shot_type_id']]
y = kobe_df1['shot_made_flag']
names=X.dtypes.index

clf = RandomForestClassifier(n_jobs=-1, n_estimators=50)

scores = []
for i in range(X.shape[1]):
     score = cross_val_score(clf, X, y, scoring="roc_auc", cv=10)
     scores.append((round(np.mean(score), 3), names[i]))
print (sorted(scores, reverse=True))
```

```
[(0.671, 'shot_area_id'), (0.67, 'game_event_id'), (0.67, 'action_type_
id'), (0.668, 'two_pt_fg'), (0.668, 'shot_range_id'), (0.667, 'season_i
d'), (0.667, 'loc_y'), (0.667, 'home_game'), (0.666, 'opponent_id'),
(0.663, 'shot_distance'), (0.662, 'loc_x'), (0.661, 'shot_zone_id'),
(0.66, 'period'), (0.657, 'time_remaining_sec'), (0.656, 'shot_type_i
d'), (0.654, 'playoffs')]
```

In [65]:
```
# Training model with Top 5 Features from above
clf = RandomForestClassifier(n_jobs=-1, n_estimators=70)

train_y = kobe_df1['shot_made_flag']
train_X = kobe_df1[['shot_area_id','game_event_id', 'action_type_id', 't
wo_pt_fg', 'shot_range_id']]
cv = StratifiedKFold(train_y, n_folds=10)

acc = cross_val_score(clf, train_X, y=train_y, cv=cv)
print "Average Accuracy On Training Data= ", acc.mean()*100, "+-", acc.s
td()*100

clf.fit(train_X, train_y)

# Predict for Era 2

test_X = kobe_df2_pre[['shot_area_id','game_event_id', 'action_type_id',
'two_pt_fg', 'shot_range_id', 'shot_id']]
predictions = clf.predict_proba(test_X.drop('shot_id', 1))
submission_b2 = pd.DataFrame({'shot_id': test_X.shot_id, 'shot_made_fla
g': predictions[:, 1]})

sub = [submission_bench, submission_b2]
submission_bench = pd.concat(sub)

submission_b2.head(10)
```

```
Average Accuracy On Training Data=  57.6641341115 +- 4.16049824255
```

Out[65]:

	shot_id	shot_made_flag
25023	25024	0.014286
25025	25026	0.257143
25026	25027	0.814286
25033	25034	0.871429
25049	25050	0.742857
25056	25057	0.542857
25066	25067	0.342857
25074	25075	0.028571
25080	25081	0.057143
25084	25085	0.571667

Model 3: Start of Phil Jackson Era

***Top 5 features will vary based run**

In [22]:
```
# Feature Selection based off of Random Forest

X = kobe_df2[['game_event_id', 'loc_x', 'loc_y', 'period', 'playoffs',
'shot_distance', 'time_remaining_sec', 'home_game', 'two_pt_fg', 'season
_id', 'action_type_id', 'shot_area_id', 'shot_zone_id', 'opponent_id',
'shot_range_id', 'shot_type_id']]
y =  kobe_df2['shot_made_flag']
names=X.dtypes.index

clf = RandomForestClassifier(n_jobs=-1, n_estimators=50)

scores = []
for i in range(X.shape[1]):
     score = cross_val_score(clf, X, y, scoring="roc_auc", cv=10)
     scores.append((round(np.mean(score), 3), names[i]))
print (sorted(scores, reverse=True))
```

```
[(0.655, 'action_type_id'), (0.651, 'shot_range_id'), (0.647, 'opponent
_id'), (0.646, 'loc_x'), (0.645, 'shot_distance'), (0.644, 'season_i
d'), (0.643, 'two_pt_fg'), (0.642, 'home_game'), (0.641, 'shot_type_i
d'), (0.64, 'loc_y'), (0.638, 'period'), (0.637, 'shot_area_id'), (0.63
5, 'shot_zone_id'), (0.634, 'playoffs'), (0.632, 'time_remaining_sec'),
(0.632, 'game_event_id')]
```

In [66]:
```
# Training model with Top 5 Features from above
clf = RandomForestClassifier(n_jobs=-1, n_estimators=70)

train_y = kobe_df2['shot_made_flag']
train_X = kobe_df2[['action_type_id', 'shot_range_id', 'opponent_id', 'l
oc_x', 'shot_distance']]
cv = StratifiedKFold(train_y, n_folds=10)

acc = cross_val_score(clf, train_X, y=train_y, cv=cv)
print "Average Accuracy On Training Data= ", acc.mean()*100, "+-", acc.s
td()*100

clf.fit(train_X, train_y)

# Predict for Era 3

test_X = kobe_df3_pre[['action_type_id', 'shot_range_id', 'opponent_id',
'loc_x', 'shot_distance', 'shot_id']]
predictions = clf.predict_proba(test_X.drop('shot_id', 1))
submission_b3 = pd.DataFrame({'shot_id': test_X.shot_id, 'shot_made_fla
g': predictions[:, 1]})

sub = [submission_bench, submission_b3]
submission_bench = pd.concat(sub)

submission_b3.head(10)
```

```
Average Accuracy On Training Data=  60.9781312835 +- 3.96792621554
```

Out[66]:

	shot_id	shot_made_flag
0	1	0.528571
7	8	0.328571
16	17	0.308690
19	20	0.308690
32	33	0.485714
33	34	0.585714
34	35	0.467143
35	36	0.814286
36	37	0.678044
37	38	0.200000

Model 4: Triangle Era

***Top 5 features will vary based run**

In [24]:
```
# Feature Selection based off of Random Forest

X = kobe_df3[['game_event_id', 'loc_x', 'loc_y', 'period', 'playoffs',
'shot_distance', 'time_remaining_sec', 'home_game', 'two_pt_fg', 'season
_id', 'action_type_id', 'shot_area_id', 'shot_zone_id', 'opponent_id',
'shot_range_id', 'shot_type_id']]
y =  kobe_df3['shot_made_flag']
names=X.dtypes.index

clf = RandomForestClassifier(n_jobs=-1, n_estimators=50)

scores = []
for i in range(X.shape[1]):
     score = cross_val_score(clf, X, y, scoring="roc_auc", cv=10)
     scores.append((round(np.mean(score), 3), names[i]))
print (sorted(scores, reverse=True))
```

```
[(0.675, 'period'), (0.674, 'playoffs'), (0.673, 'shot_type_id'), (0.67
3, 'opponent_id'), (0.673, 'loc_x'), (0.673, 'game_event_id'), (0.671,
'two_pt_fg'), (0.671, 'shot_area_id'), (0.671, 'action_type_id'), (0.6
7, 'shot_zone_id'), (0.67, 'season_id'), (0.67, 'loc_y'), (0.67, 'home_
game'), (0.668, 'shot_range_id'), (0.665, 'time_remaining_sec'), (0.66
5, 'shot_distance')]
```

In [67]:
```
# Training model with Top 5 Features from above
clf = RandomForestClassifier(n_jobs=-1, n_estimators=70)

train_y = kobe_df3['shot_made_flag']
train_X = kobe_df3[['period', 'playoffs', 'shot_type_id', 'opponent_id',
'game_event_id']]
cv = StratifiedKFold(train_y, n_folds=10)

acc = cross_val_score(clf, train_X, y=train_y, cv=cv)
print "Average Accuracy On Training Data= ", acc.mean()*100, "+-", acc.s
td()*100

clf.fit(train_X, train_y)

# Predict for Era 4

test_X = kobe_df4_pre[['period', 'playoffs', 'shot_type_id', 'opponent_i
d', 'game_event_id', 'shot_id']]
predictions = clf.predict_proba(test_X.drop('shot_id', 1))
submission_b4 = pd.DataFrame({'shot_id': test_X.shot_id, 'shot_made_fla
g': predictions[:, 1]})

sub = [submission_bench, submission_b4]
submission_bench = pd.concat(sub)

submission_b4.head(10)
```

```
Average Accuracy On Training Data=  54.6478693189 +- 1.8054036274
```

Out[67]:

	shot_id	shot_made_flag
3113	3114	0.457143
3117	3118	0.081429
3121	3122	0.646667
3131	3132	0.757143
3137	3138	0.300476
3140	3141	0.642857
3155	3156	0.628571
3161	3162	0.900000
3166	3167	0.400000
3169	3170	0.571429

Model 5: Mike D'Antoni Era

***Top 5 features will vary based run**

In [32]:
```
# Feature Selection based off of Random Forest

X = kobe_df4[['game_event_id', 'loc_x', 'loc_y', 'period', 'playoffs',
'shot_distance', 'time_remaining_sec', 'home_game', 'two_pt_fg', 'season
_id', 'action_type_id', 'shot_area_id', 'shot_zone_id', 'opponent_id',
'shot_range_id', 'shot_type_id']]
y =  kobe_df4['shot_made_flag']
names=X.dtypes.index

clf = RandomForestClassifier(n_jobs=-1, n_estimators=50)

scores = []
for i in range(X.shape[1]):
     score = cross_val_score(clf, X, y, scoring="roc_auc", cv=10)
     scores.append((round(np.mean(score), 3), names[i]))
print (sorted(scores, reverse=True))
```

```
[(0.634, 'season_id'), (0.632, 'shot_distance'), (0.632, 'loc_y'), (0.6
31, 'shot_type_id'), (0.629, 'two_pt_fg'), (0.629, 'period'), (0.629,
'game_event_id'), (0.628, 'home_game'), (0.627, 'time_remaining_sec'),
(0.626, 'playoffs'), (0.624, 'shot_area_id'), (0.621, 'action_type_i
d'), (0.62, 'shot_zone_id'), (0.62, 'shot_range_id'), (0.617, 'loc_x'),
(0.613, 'opponent_id')]
```

In [68]:
```
# Training model with Top 5 Features from above
clf = RandomForestClassifier(n_jobs=-1, n_estimators=70)

train_y = kobe_df4['shot_made_flag']
train_X = kobe_df4[['season_id', 'shot_distance', 'loc_y', 'shot_type_i
d', 'two_pt_fg']]
cv = StratifiedKFold(train_y, n_folds=10)

acc = cross_val_score(clf, train_X, y=train_y, cv=cv)
print "Average Accuracy On Training Data= ", acc.mean()*100, "+-", acc.s
td()*100

clf.fit(train_X, train_y)

# Predict for Era 5

test_X = kobe_df5_pre[['season_id', 'shot_distance', 'loc_y', 'shot_type
_id', 'two_pt_fg', 'shot_id']]
predictions = clf.predict_proba(test_X.drop('shot_id', 1))
submission_b5 = pd.DataFrame({'shot_id': test_X.shot_id, 'shot_made_fla
g': predictions[:, 1]})

sub = [submission_bench, submission_b5]
submission_bench = pd.concat(sub)

submission_b5.head(10)
```

```
Average Accuracy On Training Data=  57.3406735109 +- 2.44214062339
```

Out[68]:

	shot_id	shot_made_flag
19417	19418	0.528571
19428	19429	1.000000
19445	19446	0.728571
19457	19458	0.696259
19459	19460	0.064286
19461	19462	0.128571
19464	19465	0.895238
19471	19472	0.526789
19479	19480	0.095238
19502	19503	0.822670

Model 6: Injury/Coach Transition Era

***Top 5 features will vary based run**

In [34]:
```
# Feature Selection based off of Random Forest

X = kobe_df5[['game_event_id', 'loc_x', 'loc_y', 'period', 'playoffs',
'shot_distance', 'time_remaining_sec', 'home_game', 'two_pt_fg', 'season
_id', 'action_type_id', 'shot_area_id', 'shot_zone_id', 'opponent_id',
'shot_range_id', 'shot_type_id']]
y =  kobe_df5['shot_made_flag']
names=X.dtypes.index

clf = RandomForestClassifier(n_jobs=-1, n_estimators=50)

scores = []
for i in range(X.shape[1]):
     score = cross_val_score(clf, X, y, scoring="roc_auc", cv=10)
     scores.append((round(np.mean(score), 3), names[i]))
print (sorted(scores, reverse=True))
```

```
[(0.688, 'loc_y'), (0.686, 'loc_x'), (0.684, 'season_id'), (0.684, 'pla
yoffs'), (0.683, 'two_pt_fg'), (0.682, 'shot_type_id'), (0.682, 'shot_r
ange_id'), (0.682, 'game_event_id'), (0.68, 'shot_area_id'), (0.68, 'op
ponent_id'), (0.679, 'shot_zone_id'), (0.678, 'time_remaining_sec'),
(0.677, 'shot_distance'), (0.675, 'period'), (0.675, 'action_type_id'),
(0.672, 'home_game')]
```

In [69]:
```
# Training model with Top 5 Features from above
clf = RandomForestClassifier(n_jobs=-1, n_estimators=70)

train_y = kobe_df5['shot_made_flag']
train_X = kobe_df5[['loc_y', 'loc_x', 'season_id', 'playoffs', 'two_pt_f
g']]
cv = StratifiedKFold(train_y, n_folds=10)

acc = cross_val_score(clf, train_X, y=train_y, cv=cv)
print "Average Accuracy On Training Data= ", acc.mean()*100, "+-", acc.s
td()*100

clf.fit(train_X, train_y)

# Predict for Era 6

test_X = kobe_df6_pre[['loc_y', 'loc_x', 'season_id', 'playoffs', 'two_p
t_fg', 'shot_id']]
predictions = clf.predict_proba(test_X.drop('shot_id', 1))
submission_b6 = pd.DataFrame({'shot_id': test_X.shot_id, 'shot_made_fla
g': predictions[:, 1]})

sub = [submission_bench, submission_b6]
submission_bench = pd.concat(sub)

submission_b6.head(10)
```

```
Average Accuracy On Training Data=  58.259923738 +- 5.56440363426
```

Out[69]:

	shot_id	shot_made_flag
21078	21079	0.485714
21089	21090	0.471429
21092	21093	0.028571
21103	21104	0.257143
21113	21114	0.185714
21116	21117	0.785714
21119	21120	0.185714
21120	21121	0.185714
21131	21132	0.300000
21135	21136	0.785714

Kaggle Submission

In [79]:
```
submission_bench.sort_values(['shot_id'], ascending=[True], inplace=Tru
e)
submission_bench.to_csv('submission_bench.csv')
```

We submitted our CSV file from above to the Kaggle for scoring. Our Benchmark Predictions received a score of 1.04202.

According to the Evaluation Page for this Kaggle Competition, our score is generated by running our predictions through a Log-Loss function.

Attempting To Surpass our Benchmark

Columns we will cluster using K-Means:

Shot Location:

X1 = kobe_df0[['loc_x','loc_y', 'shot_distance']]

Where Factor:

X2 = kobe_df0[['season_id','game_event_id','action_type_id']]

Shot Location

On the shot location we see two distinctive clusters, within the three-point line and beyond the three-point line. While some shot do occur in the apparent void, this is to be expected to be the nature of the area. The difference of one inch of foot placement can be the difference between two and three points, so players often make a conscious effort to establish themselves beyond the three-point line when making long-range jump shots.

Where Factor

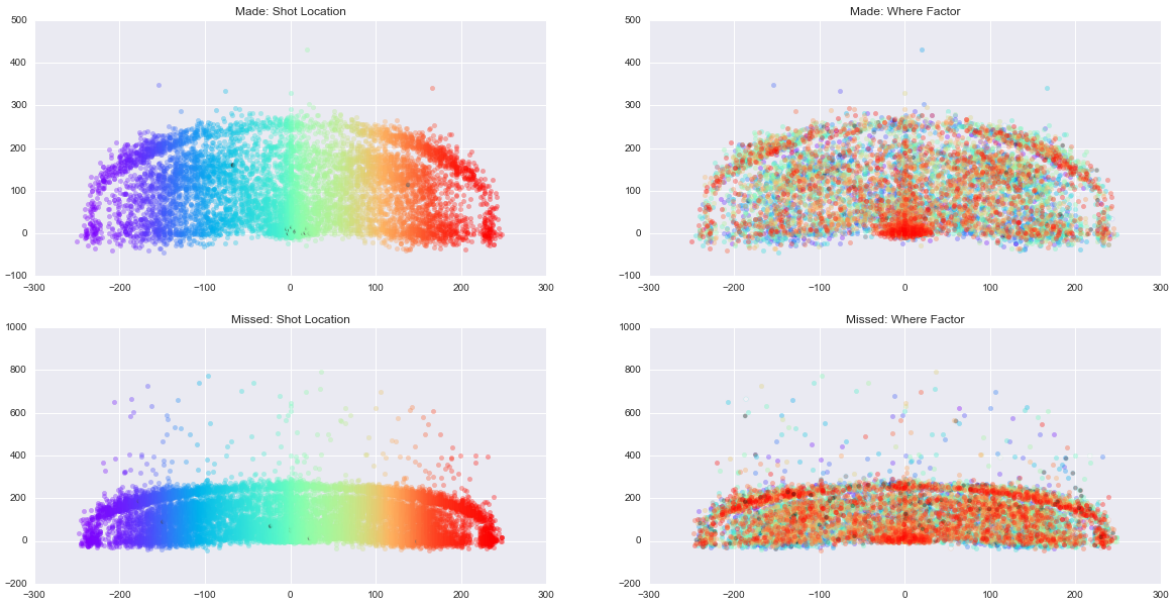
In this chart we see the same cluster as shown the shot chart. Each dot color represents a specific type of action which bring s our attention to the area directly around the basket which in entirely red/orange. This cluster of shot type is directly corresponding to the distance, dunks and layups will almost always occur within a few feet of the basket (depends if how they count floaters).

Makes vs. Misses

The biggest thing that we notice when comparing the charts is that they look nearly identical. Both appear to have a cluster within and beyond the three-point line, as well as a dunk/lay up cluster around where the basket would be. The only major difference is that the misses have a large amount of outliers that were shot from far distances (more than likely a rushed shot, attempted before the clock expired). These outliers do not form a cluster and seem to provide noise for our visuals.

In [15]:
```
plt.figure(figsize=(20,10))

h1 = df.loc[df.shot_made_flag == 1]
h2 = df.loc[df.shot_made_flag == 0]

# Shot Location
plt.subplot(221)
scatter_plot_by_category(h1, ('loc_x','loc_y', 'shot_distance'))
plt.title('Made: Shot Location')

# Where Factor
plt.subplot(222)
scatter_plot_by_category(h1, ('season_id','game_event_id','action_type_i
d'))
plt.title('Made: Where Factor')

# Shot Location
plt.subplot(223)
scatter_plot_by_category(h2, ('loc_x','loc_y', 'shot_distance'))
plt.title('Missed: Shot Location')

# Where Factor
plt.subplot(224)
scatter_plot_by_category(h2, ('season_id','game_event_id','action_type_i
d'))
plt.title('Missed: Where Factor')
```

Out[15]: <matplotlib.text.Text at 0x11b43c6d0>



Using K-Means and Random Forest

We chose K-Means as our clustering method of choice because of the clustering and central mean. The original plan was to use the DBScan techniques to develop our clusters because of location advantages of the method, but the additives to the clusters suggest multiple dimensions that require a central mean to create the best cluster.

Model 0: Rookie Season

In [17]:
```
# Randomly choosing a value for Rookie season missing data

submission = pd.DataFrame({'shot_id': kobe_df0_pre.shot_id})
submission['shot_made_flag'] = np.random.uniform(0.01, 0.99, len(submiss
ion))
submission.head(10)
```

Out[17]:

	shot_id	shot_made_flag
22906	22907	0.251235
22908	22909	0.219769
22925	22926	0.145133
22926	22927	0.742689
22929	22930	0.299266
22933	22934	0.751327
22936	22937	0.977931
22938	22939	0.155168
22948	22949	0.356700
22950	22951	0.676945

Model 1: Adjusting to the league

Finding Our Parameters

In [25]:
```
# 'period', 'playoffs', 'time_remaining_sec', 'home_game',
# 'two_pt_fg', 'shot_area_id', 'shot_zone_id', 'opponent_id', 'shot_rang
e_id'

from sklearn.cluster import KMeans

clf = RandomForestClassifier(n_jobs=-1, n_estimators=70)
cv = StratifiedKFold(kobe_df0['shot_made_flag'], n_folds=10)

# Shot Location
X1 = kobe_df0[['loc_x','loc_y', 'shot_distance']]

# Where Factor
X2 = kobe_df0[['season_id','game_event_id','action_type_id']]

params = []
for n_shot in range(2,5):
    for n_where in range(2,7):
        # get the first clustering
        cls_shot = KMeans(n_clusters=n_shot, init='k-means++',random_sta
te=1)
        cls_shot.fit(X1)
        newfeature_shot = cls_shot.labels_ # the labels from kmeans clus
tering

        # append on the second clustering
        cls_where = KMeans(n_clusters=n_where, init='k-means++',random_s
tate=1)
        cls_where.fit(X2)
        newfeature_where = cls_where.labels_ # the labels from kmeans cl
ustering

        y = kobe_df0['shot_made_flag']
        X = kobe_df0[['home_game', 'time_remaining_sec']]
        X = np.column_stack((X, pd.get_dummies(newfeature_shot),pd.get_d
ummies(newfeature_where)))

        acc = cross_val_score(clf,X,y=y,cv=cv)
        params.append((n_shot,n_where,acc.mean()*100,acc.std()*100)) # s
ave state

        print "Clusters",n_shot,n_where,"Average accuracy = ", acc.mean
()*100, "+-", acc.std()*100
```

```
Clusters 2 2 Average accuracy =  48.0560053981 +- 7.22560379898
Clusters 2 3 Average accuracy =  50.9308367072 +- 5.87783526094
Clusters 2 4 Average accuracy =  50.4773954116 +- 7.76948892188
Clusters 2 5 Average accuracy =  51.7267206478 +- 6.6803094696
Clusters 2 6 Average accuracy =  49.6747638327 +- 9.08641386613
Clusters 3 2 Average accuracy =  53.266194332 +- 3.07275197384
Clusters 3 3 Average accuracy =  51.963900135 +- 3.38753569679
Clusters 3 4 Average accuracy =  52.5300269906 +- 6.26033768365
Clusters 3 5 Average accuracy =  50.1680161943 +- 5.62600137968
Clusters 3 6 Average accuracy =  50.1481106613 +- 6.81716949856
Clusters 4 2 Average accuracy =  54.0226045884 +- 4.90139828798
Clusters 4 3 Average accuracy =  55.0951417004 +- 5.78104515148
Clusters 4 4 Average accuracy =  56.1676788124 +- 2.9078299137
Clusters 4 5 Average accuracy =  54.0819838057 +- 7.53467570903
Clusters 4 6 Average accuracy =  53.2857624831 +- 5.53547692047
```

Building Our Model

In [26]:
```
clf = RandomForestClassifier(n_estimators=70,random_state=1)

n_shot, n_where = 4, 4

# Shot Location
X1 = kobe_df0[['loc_x','loc_y', 'shot_distance']]

# Where Factor
X2 = kobe_df0[['season_id','game_event_id','action_type_id']]

# get the first clustering
cls_shot = KMeans(n_clusters=n_shot, init='k-means++',random_state=1)
cls_shot.fit(X1)
newfeature_shot = cls_shot.labels_ # the labels from kmeans clustering

# append on the second clustering
cls_where = KMeans(n_clusters=n_where, init='k-means++',random_state=1)
cls_where.fit(X2)
newfeature_where = cls_where.labels_ # the labels from kmeans clustering

y = kobe_df0['shot_made_flag']
X = kobe_df0[['home_game', 'time_remaining_sec']]
X = np.column_stack((X, pd.get_dummies(newfeature_shot),pd.get_dummies(n
ewfeature_where)))

cv = StratifiedKFold(y, n_folds=10)
acc = cross_val_score(clf,X,y=y,cv=cv)

clf.fit(X, y)

print "Clusters",n_shot,n_where,"Average accuracy = ", acc.mean()*100,
"+-", acc.std()*100
```

```
Clusters 4 4 Average accuracy =  56.161268556 +- 3.00407472771
```

Prediction

In [27]:
```
# Shot Location
test_X1 = kobe_df1_pre[['loc_x','loc_y', 'shot_distance']]
newfeature_shot = cls_shot.predict(test_X1)

# Where Factor
test_X2 = kobe_df1_pre[['season_id','game_event_id','action_type_id']]
newfeature_where = cls_where.predict(test_X2)

test_X = kobe_df1_pre[['home_game', 'time_remaining_sec', 'shot_id']]

predictions = clf.predict_proba(np.column_stack((test_X.drop('shot_id',
1), pd.get_dummies(newfeature_shot),pd.get_dummies(newfeature_where))))
submission_1 = pd.DataFrame({'shot_id': test_X.shot_id, 'shot_made_fla
g': predictions[:, 1]})

sub = [submission, submission_1]
submission = pd.concat(sub)

submission_1.head(10)
```

Out[27]:

	shot_id	shot_made_flag
23341	23342	0.614286
23343	23344	0.000000
23349	23350	0.128571
23366	23367	0.471429
23367	23368	0.042857
23373	23374	0.242857
23386	23387	0.057143
23388	23389	0.228571
23398	23399	0.971429
23402	23403	0.042857

Model 2: Coach/Philosopy Transition

Finding Our Parameters

In [32]:
```
# 'period', 'playoffs', 'time_remaining_sec', 'home_game',
# 'two_pt_fg', 'shot_area_id', 'shot_zone_id', 'opponent_id', 'shot_rang
e_id'

cv = StratifiedKFold(kobe_df1['shot_made_flag'], n_folds=10)

# Shot Location
X1 = kobe_df1[['loc_x','loc_y', 'shot_distance']]

# Where Factor
X2 = kobe_df1[['season_id','game_event_id','action_type_id']]

params = []
for n_shot in range(2,7):
    for n_where in range(3,7):
        # get the first clustering
        cls_shot = KMeans(n_clusters=n_shot, init='k-means++',random_sta
te=1)
        cls_shot.fit(X1)
        newfeature_shot = cls_shot.labels_ # the labels from kmeans clus
tering

        # append on the second clustering
        cls_where = KMeans(n_clusters=n_where, init='k-means++',random_s
tate=1)
        cls_where.fit(X2)
        newfeature_where = cls_where.labels_ # the labels from kmeans cl
ustering

        y = kobe_df1['shot_made_flag']
        X = kobe_df1[['home_game', 'shot_area_id']] # Clusters 6 5 Avera
ge accuracy =  64.5664861454 +- 3.89074057912
        X = np.column_stack((X, pd.get_dummies(newfeature_shot),pd.get_d
ummies(newfeature_where)))

        acc = cross_val_score(clf,X,y=y,cv=cv)
        params.append((n_shot,n_where,acc.mean()*100,acc.std()*100)) # s
ave state

        print "Clusters",n_shot,n_where,"Average accuracy = ", acc.mean
()*100, "+-", acc.std()*100
```

```
Clusters 2 3 Average accuracy =  61.5741714426 +- 5.09724656459
Clusters 2 4 Average accuracy =  60.2584696006 +- 5.36132313702
Clusters 2 5 Average accuracy =  58.4450198924 +- 5.1539536413
Clusters 2 6 Average accuracy =  57.0063270063 +- 5.58818710944
Clusters 3 3 Average accuracy =  63.2456578509 +- 4.7923736626
Clusters 3 4 Average accuracy =  61.0291901081 +- 5.54158155242
Clusters 3 5 Average accuracy =  62.0818216871 +- 4.21999266147
Clusters 3 6 Average accuracy =  57.7805089647 +- 4.26885899928
Clusters 4 3 Average accuracy =  63.645389698 +- 5.49253265633
Clusters 4 4 Average accuracy =  62.7210070631 +- 6.07173957819
Clusters 4 5 Average accuracy =  63.5104807473 +- 4.55005649401
Clusters 4 6 Average accuracy =  58.6862698705 +- 5.93448418418
Clusters 5 3 Average accuracy =  63.3906882591 +- 3.53009736023
Clusters 5 4 Average accuracy =  61.9467374731 +- 4.27349881185
Clusters 5 5 Average accuracy =  62.3499745868 +- 3.94005219039
Clusters 5 6 Average accuracy =  58.8363829153 +- 4.02830756496
Clusters 6 3 Average accuracy =  64.6861471861 +- 4.85660714081
Clusters 6 4 Average accuracy =  64.173414305 +- 4.64864945105
Clusters 6 5 Average accuracy =  64.5664861454 +- 3.89074057912
Clusters 6 6 Average accuracy =  59.8755192176 +- 4.39647084365
```

Building Our Model

In [33]:
```
n_shot, n_where = 6, 5

# Shot Location
X1 = kobe_df1[['loc_x','loc_y', 'shot_distance']]

# Where Factor
X2 = kobe_df1[['season_id','game_event_id','action_type_id']]

# get the first clustering
cls_shot = KMeans(n_clusters=n_shot, init='k-means++',random_state=1)
cls_shot.fit(X1)
newfeature_shot = cls_shot.labels_ # the labels from kmeans clustering

# append on the second clustering
cls_where = KMeans(n_clusters=n_where, init='k-means++',random_state=1)
cls_where.fit(X2)
newfeature_where = cls_where.labels_ # the labels from kmeans clustering

y = kobe_df1['shot_made_flag']
X = kobe_df1[['home_game', 'shot_area_id']]
X = np.column_stack((X, pd.get_dummies(newfeature_shot),pd.get_dummies(n
ewfeature_where)))

cv = StratifiedKFold(y, n_folds=10)
acc = cross_val_score(clf,X,y=y,cv=cv)

clf.fit(X, y)

print "Clusters",n_shot,n_where,"Average accuracy = ", acc.mean()*100,
"+-", acc.std()*100
```

```
Clusters 6 5 Average accuracy =  64.5664861454 +- 3.89074057912
```

Prediction

In [34]:
```
# Shot Location
test_X1 = kobe_df2_pre[['loc_x','loc_y', 'shot_distance']]
newfeature_shot = cls_shot.predict(test_X1)

# Where Factor
test_X2 = kobe_df2_pre[['season_id','game_event_id','action_type_id']]
newfeature_where = cls_where.predict(test_X2)

test_X = kobe_df2_pre[['home_game', 'shot_area_id', 'shot_id']]

predictions = clf.predict_proba(np.column_stack((test_X.drop('shot_id',
1), pd.get_dummies(newfeature_shot),pd.get_dummies(newfeature_where))))
submission_2 = pd.DataFrame({'shot_id': test_X.shot_id, 'shot_made_fla
g': predictions[:, 1]})

sub = [submission, submission_2]
submission = pd.concat(sub)

submission_2.head(10)
```

Out[34]:

	shot_id	shot_made_flag
25023	25024	0.243449
25025	25026	0.112799
25026	25027	0.688369
25033	25034	0.285998
25049	25050	0.378047
25056	25057	0.743673
25066	25067	0.243449
25074	25075	0.361996
25080	25081	0.222629
25084	25085	0.188907

Model 3: Start of Phil Jackson Era

Finding Our Parameters

In [39]:
```
# 'period', 'playoffs', 'time_remaining_sec', 'home_game',
# 'two_pt_fg', 'shot_area_id', 'shot_zone_id', 'opponent_id', 'shot_rang
e_id'

cv = StratifiedKFold(kobe_df2['shot_made_flag'], n_folds=10)

# Shot Location
X1 = kobe_df2[['loc_x','loc_y', 'shot_distance']]

# Where Factor
X2 = kobe_df2[['season_id','game_event_id','action_type_id']]

params = []
for n_shot in range(2,5):
    for n_where in range(3,7):
        # get the first clustering
        cls_shot = KMeans(n_clusters=n_shot, init='k-means++',random_sta
te=1)
        cls_shot.fit(X1)
        newfeature_shot = cls_shot.labels_ # the labels from kmeans clus
tering

        # append on the second clustering
        cls_where = KMeans(n_clusters=n_where, init='k-means++',random_s
tate=1)
        cls_where.fit(X2)
        newfeature_where = cls_where.labels_ # the labels from kmeans cl
ustering

        y = kobe_df2['shot_made_flag']
        X = kobe_df2[['home_game', 'shot_zone_id']] # Clusters 2 5 Avera
ge accuracy =  59.6798341608 +- 3.15829796528
        X = np.column_stack((X, pd.get_dummies(newfeature_shot),pd.get_d
ummies(newfeature_where)))

        acc = cross_val_score(clf,X,y=y,cv=cv)
        params.append((n_shot,n_where,acc.mean()*100,acc.std()*100)) # s
ave state

        print "Clusters",n_shot,n_where,"Average accuracy = ", acc.mean
()*100, "+-", acc.std()*100
```

```
Clusters 2 3 Average accuracy =  59.5219132013 +- 3.49106426564
Clusters 2 4 Average accuracy =  59.900718874 +- 3.8306382013
Clusters 2 5 Average accuracy =  59.6798341608 +- 3.15829796528
Clusters 2 6 Average accuracy =  58.1449047136 +- 3.68164476126
Clusters 3 3 Average accuracy =  58.8389384153 +- 4.78406800511
Clusters 3 4 Average accuracy =  58.9857737682 +- 4.78784073458
Clusters 3 5 Average accuracy =  59.4426324312 +- 5.11618050494
Clusters 3 6 Average accuracy =  59.8243207175 +- 3.67983022932
Clusters 4 3 Average accuracy =  56.936867204 +- 4.17968489587
Clusters 4 4 Average accuracy =  59.9793679603 +- 4.40607315114
Clusters 4 5 Average accuracy =  58.3774266446 +- 3.41123552051
Clusters 4 6 Average accuracy =  58.8342675137 +- 4.06772703961
```

Building Our Model

In [40]:
```
n_shot, n_where = 2, 5

# Shot Location
X1 = kobe_df2[['loc_x','loc_y', 'shot_distance']]

# Where Factor
X2 = kobe_df2[['season_id','game_event_id','action_type_id']]

# get the first clustering
cls_shot = KMeans(n_clusters=n_shot, init='k-means++',random_state=1)
cls_shot.fit(X1)
newfeature_shot = cls_shot.labels_ # the labels from kmeans clustering

# append on the second clustering
cls_where = KMeans(n_clusters=n_where, init='k-means++',random_state=1)
cls_where.fit(X2)
newfeature_where = cls_where.labels_ # the labels from kmeans clustering

y = kobe_df2['shot_made_flag']
X = kobe_df2[['home_game', 'shot_zone_id']]
X = np.column_stack((X, pd.get_dummies(newfeature_shot),pd.get_dummies(n
ewfeature_where)))

cv = StratifiedKFold(y, n_folds=10)
acc = cross_val_score(clf,X,y=y,cv=cv)

clf.fit(X, y)

print "Clusters",n_shot,n_where,"Average accuracy = ", acc.mean()*100,
"+-", acc.std()*100
```

```
Clusters 2 5 Average accuracy =  59.6798341608 +- 3.15829796528
```

Prediction

In [41]:
```
# Shot Location
test_X1 = kobe_df3_pre[['loc_x','loc_y', 'shot_distance']]
newfeature_shot = cls_shot.predict(test_X1)

# Where Factor
test_X2 = kobe_df3_pre[['season_id','game_event_id','action_type_id']]
newfeature_where = cls_where.predict(test_X2)

test_X = kobe_df3_pre[['home_game', 'shot_zone_id', 'shot_id']]

predictions = clf.predict_proba(np.column_stack((test_X.drop('shot_id',
1), pd.get_dummies(newfeature_shot),pd.get_dummies(newfeature_where))))
submission_3 = pd.DataFrame({'shot_id': test_X.shot_id, 'shot_made_fla
g': predictions[:, 1]})

sub = [submission, submission_3]
submission = pd.concat(sub)

submission_3.head(10)
```

Out[41]:

	shot_id	shot_made_flag
0	1	0.386959
7	8	0.656830
16	17	0.801773
19	20	0.639823
32	33	0.386959
33	34	0.386959
34	35	0.704811
35	36	0.704811
36	37	0.704811
37	38	0.287799

Model 4: Triangle Era

Finding Our Parameters

In [45]:
```
# 'period', 'playoffs', 'time_remaining_sec', 'home_game',
# 'two_pt_fg', 'shot_area_id', 'shot_zone_id', 'opponent_id', 'shot_rang
e_id'

cv = StratifiedKFold(kobe_df3['shot_made_flag'], n_folds=10)

# Shot Location
X1 = kobe_df3[['loc_x','loc_y', 'shot_distance']]

# Where Factor
X2 = kobe_df3[['season_id','game_event_id','action_type_id']]

params = []
for n_shot in range(4,7):
    for n_where in range(3,7):
        # get the first clustering
        cls_shot = KMeans(n_clusters=n_shot, init='k-means++',random_sta
te=1)
        cls_shot.fit(X1)
        newfeature_shot = cls_shot.labels_ # the labels from kmeans clus
tering

        # append on the second clustering
        cls_where = KMeans(n_clusters=n_where, init='k-means++',random_s
tate=1)
        cls_where.fit(X2)
        newfeature_where = cls_where.labels_ # the labels from kmeans cl
ustering

        y = kobe_df3['shot_made_flag']
        X = kobe_df3[['home_game', 'shot_zone_id']] # Clusters 4 3 Avera
ge accuracy =  60.5210517145 +- 1.60809685049
        X = np.column_stack((X, pd.get_dummies(newfeature_shot),pd.get_d
ummies(newfeature_where)))

        acc = cross_val_score(clf,X,y=y,cv=cv)
        params.append((n_shot,n_where,acc.mean()*100,acc.std()*100)) # s
ave state

        print "Clusters",n_shot,n_where,"Average accuracy = ", acc.mean
()*100, "+-", acc.std()*100
```

```
Clusters 4 3 Average accuracy =  60.5210517145 +- 1.60809685049
Clusters 4 4 Average accuracy =  60.4992658386 +- 1.92243818723
Clusters 4 5 Average accuracy =  59.9122121272 +- 1.65132681483
Clusters 4 6 Average accuracy =  59.6066810746 +- 1.39670091116
Clusters 5 3 Average accuracy =  59.8465211165 +- 1.52935183628
Clusters 5 4 Average accuracy =  59.9337146535 +- 1.68543450789
Clusters 5 5 Average accuracy =  60.1298869868 +- 1.67812473328
Clusters 5 6 Average accuracy =  58.7798336558 +- 1.4916740435
Clusters 6 3 Average accuracy =  59.9551698177 +- 1.58526414305
Clusters 6 4 Average accuracy =  59.2813959834 +- 1.491977777
Clusters 6 5 Average accuracy =  58.9323864644 +- 1.5353273729
Clusters 6 6 Average accuracy =  58.1270423938 +- 1.42955683792
```

Building Our Model

In [46]:
```
n_shot, n_where = 4, 3

# Shot Location
X1 = kobe_df3[['loc_x','loc_y', 'shot_distance']]

# Where Factor
X2 = kobe_df3[['season_id','game_event_id','action_type_id']]

# get the first clustering
cls_shot = KMeans(n_clusters=n_shot, init='k-means++',random_state=1)
cls_shot.fit(X1)
newfeature_shot = cls_shot.labels_ # the labels from kmeans clustering

# append on the second clustering
cls_where = KMeans(n_clusters=n_where, init='k-means++',random_state=1)
cls_where.fit(X2)
newfeature_where = cls_where.labels_ # the labels from kmeans clustering

y = kobe_df3['shot_made_flag']
X = kobe_df3[['home_game', 'shot_zone_id']]
X = np.column_stack((X, pd.get_dummies(newfeature_shot),pd.get_dummies(n
ewfeature_where)))

cv = StratifiedKFold(y, n_folds=10)
acc = cross_val_score(clf,X,y=y,cv=cv)

clf.fit(X, y)

print "Clusters",n_shot,n_where,"Average accuracy = ", acc.mean()*100,
"+-", acc.std()*100
```

```
Clusters 4 3 Average accuracy =  60.5210517145 +- 1.60809685049
```

Prediction

In [47]:
```
# Shot Location
test_X1 = kobe_df4_pre[['loc_x','loc_y', 'shot_distance']]
newfeature_shot = cls_shot.predict(test_X1)

# Where Factor
test_X2 = kobe_df4_pre[['season_id','game_event_id','action_type_id']]
newfeature_where = cls_where.predict(test_X2)

test_X = kobe_df4_pre[['home_game', 'shot_zone_id', 'shot_id']]

predictions = clf.predict_proba(np.column_stack((test_X.drop('shot_id',
1), pd.get_dummies(newfeature_shot),pd.get_dummies(newfeature_where))))
submission_4 = pd.DataFrame({'shot_id': test_X.shot_id, 'shot_made_fla
g': predictions[:, 1]})

sub = [submission, submission_4]
submission = pd.concat(sub)

submission_4.head(10)
```

Out[47]:

	shot_id	shot_made_flag
3113	3114	0.657996
3117	3118	0.470280
3121	3122	0.408234
3131	3132	0.437053
3137	3138	0.413284
3140	3141	0.643467
3155	3156	0.643467
3161	3162	0.631066
3166	3167	0.323756
3169	3170	0.344802

Model 5: Mike D'Antoni Era

Finding Our Parameters

In [50]:
```
# 'period', 'playoffs', 'time_remaining_sec', 'home_game',
# 'two_pt_fg', 'shot_area_id', 'shot_zone_id', 'opponent_id', 'shot_rang
e_id'

cv = StratifiedKFold(kobe_df4['shot_made_flag'], n_folds=10)

# Shot Location
X1 = kobe_df4[['loc_x','loc_y', 'shot_distance']]

# Where Factor
X2 = kobe_df4[['season_id','game_event_id','action_type_id']]

params = []
for n_shot in range(4,7):
    for n_where in range(3,7):
        # get the first clustering
        cls_shot = KMeans(n_clusters=n_shot, init='k-means++',random_sta
te=1)
        cls_shot.fit(X1)
        newfeature_shot = cls_shot.labels_ # the labels from kmeans clus
tering

        # append on the second clustering
        cls_where = KMeans(n_clusters=n_where, init='k-means++',random_s
tate=1)
        cls_where.fit(X2)
        newfeature_where = cls_where.labels_ # the labels from kmeans cl
ustering

        y = kobe_df4['shot_made_flag']
        X = kobe_df4[['home_game', 'shot_zone_id']] # Clusters 4 3 Avera
ge accuracy =  61.0826761891 +- 1.713323951
        X = np.column_stack((X, pd.get_dummies(newfeature_shot),pd.get_d
ummies(newfeature_where)))

        acc = cross_val_score(clf,X,y=y,cv=cv)
        params.append((n_shot,n_where,acc.mean()*100,acc.std()*100)) # s
ave state

        print "Clusters",n_shot,n_where,"Average accuracy = ", acc.mean
()*100, "+-", acc.std()*100
```

```
Clusters 4 3 Average accuracy =  61.0826761891 +- 1.713323951
Clusters 4 4 Average accuracy =  59.384516193 +- 2.55679961583
Clusters 4 5 Average accuracy =  58.0379903784 +- 3.52463656336
Clusters 4 6 Average accuracy =  59.0418092546 +- 3.43571312944
Clusters 5 3 Average accuracy =  60.2415315181 +- 2.19507338029
Clusters 5 4 Average accuracy =  58.2626593265 +- 2.07790545823
Clusters 5 5 Average accuracy =  57.6313048653 +- 3.44745556733
Clusters 5 6 Average accuracy =  58.6971184844 +- 4.08863856711
Clusters 6 3 Average accuracy =  60.9497594604 +- 2.37612622229
Clusters 6 4 Average accuracy =  61.0856519367 +- 2.34917017318
Clusters 6 5 Average accuracy =  58.2636512424 +- 3.2003432823
Clusters 6 6 Average accuracy =  58.049893369 +- 3.20510049084
```

Building Our Model

In [51]:
```
n_shot, n_where = 4, 3

# Shot Location
X1 = kobe_df4[['loc_x','loc_y', 'shot_distance']]

# Where Factor
X2 = kobe_df4[['season_id','game_event_id','action_type_id']]

# get the first clustering
cls_shot = KMeans(n_clusters=n_shot, init='k-means++',random_state=1)
cls_shot.fit(X1)
newfeature_shot = cls_shot.labels_ # the labels from kmeans clustering

# append on the second clustering
cls_where = KMeans(n_clusters=n_where, init='k-means++',random_state=1)
cls_where.fit(X2)
newfeature_where = cls_where.labels_ # the labels from kmeans clustering

y = kobe_df4['shot_made_flag']
X = kobe_df4[['home_game', 'shot_zone_id']]
X = np.column_stack((X, pd.get_dummies(newfeature_shot),pd.get_dummies(n
ewfeature_where)))

cv = StratifiedKFold(y, n_folds=10)
acc = cross_val_score(clf,X,y=y,cv=cv)

clf.fit(X, y)

print "Clusters",n_shot,n_where,"Average accuracy = ", acc.mean()*100,
"+-", acc.std()*100
```

```
Clusters 4 3 Average accuracy =  61.0826761891 +- 1.713323951
```

Prediction

In [52]:
```
# Shot Location
test_X1 = kobe_df5_pre[['loc_x','loc_y', 'shot_distance']]
newfeature_shot = cls_shot.predict(test_X1)

# Where Factor
test_X2 = kobe_df5_pre[['season_id','game_event_id','action_type_id']]
newfeature_where = cls_where.predict(test_X2)

test_X = kobe_df5_pre[['home_game', 'shot_zone_id', 'shot_id']]

predictions = clf.predict_proba(np.column_stack((test_X.drop('shot_id',
1), pd.get_dummies(newfeature_shot),pd.get_dummies(newfeature_where))))
submission_5 = pd.DataFrame({'shot_id': test_X.shot_id, 'shot_made_fla
g': predictions[:, 1]})

sub = [submission, submission_5]
submission = pd.concat(sub)

submission_5.head(10)
```

Out[52]:

	shot_id	shot_made_flag
19417	19418	0.439901
19428	19429	0.687031
19445	19446	0.488329
19457	19458	0.439901
19459	19460	0.352313
19461	19462	0.613306
19464	19465	0.355531
19471	19472	0.620896
19479	19480	0.360072
19502	19503	0.620896

Model 6: Injury/Coach Transition Era

Finding Our Parameters

In [54]:
```
# 'period', 'playoffs', 'time_remaining_sec', 'home_game',
# 'two_pt_fg', 'shot_area_id', 'shot_zone_id', 'opponent_id', 'shot_rang
e_id'

cv = StratifiedKFold(kobe_df5['shot_made_flag'], n_folds=10)

# Shot Location
X1 = kobe_df5[['loc_x','loc_y', 'shot_distance']]

# Where Factor
X2 = kobe_df5[['season_id','game_event_id','action_type_id']]

params = []
for n_shot in range(2,7):
    for n_where in range(3,7):
        # get the first clustering
        cls_shot = KMeans(n_clusters=n_shot, init='k-means++',random_sta
te=1)
        cls_shot.fit(X1)
        newfeature_shot = cls_shot.labels_ # the labels from kmeans clus
tering

        # append on the second clustering
        cls_where = KMeans(n_clusters=n_where, init='k-means++',random_s
tate=1)
        cls_where.fit(X2)
        newfeature_where = cls_where.labels_ # the labels from kmeans cl
ustering

        y = kobe_df5['shot_made_flag']
        X = kobe_df5[['period', 'shot_zone_id']] # Clusters 2 4 Average
accuracy =  60.6444806887 +- 3.55802032875
        X = np.column_stack((X, pd.get_dummies(newfeature_shot),pd.get_d
ummies(newfeature_where)))

        acc = cross_val_score(clf,X,y=y,cv=cv)
        params.append((n_shot,n_where,acc.mean()*100,acc.std()*100)) # s
ave state

        print "Clusters",n_shot,n_where,"Average accuracy = ", acc.mean
()*100, "+-", acc.std()*100
```

```
Clusters 2 3 Average accuracy =  60.2143750838 +- 3.59896897128
Clusters 2 4 Average accuracy =  60.6444806887 +- 3.55802032875
Clusters 2 5 Average accuracy =  59.991807796 +- 3.44127276967
Clusters 2 6 Average accuracy =  59.7878740486 +- 5.26820541445
Clusters 3 3 Average accuracy =  59.9265904047 +- 3.61703434116
Clusters 3 4 Average accuracy =  59.9912790265 +- 3.65614086893
Clusters 3 5 Average accuracy =  59.4897448501 +- 3.82505939571
Clusters 3 6 Average accuracy =  59.1398334748 +- 5.57498348093
Clusters 4 3 Average accuracy =  57.9038384199 +- 3.70047492029
Clusters 4 4 Average accuracy =  57.9028032233 +- 2.89986744625
Clusters 4 5 Average accuracy =  57.5384214368 +- 2.93295131711
Clusters 4 6 Average accuracy =  58.0513502242 +- 4.83517526258
Clusters 5 3 Average accuracy =  58.3313820993 +- 3.65669751428
Clusters 5 4 Average accuracy =  57.818438417 +- 3.23944249344
Clusters 5 5 Average accuracy =  56.8117282571 +- 3.19391715125
Clusters 5 6 Average accuracy =  57.3251858141 +- 3.23826989415
Clusters 6 3 Average accuracy =  56.4612955598 +- 4.86391808957
Clusters 6 4 Average accuracy =  57.5332380059 +- 4.47958848216
Clusters 6 5 Average accuracy =  56.1740470977 +- 3.61291764145
Clusters 6 6 Average accuracy =  56.6797146134 +- 5.1870038721
```

Building Our Model

In [55]:
```
n_shot, n_where = 2, 4

# Shot Location
X1 = kobe_df5[['loc_x','loc_y', 'shot_distance']]

# Where Factor
X2 = kobe_df5[['season_id','game_event_id','action_type_id']]

# get the first clustering
cls_shot = KMeans(n_clusters=n_shot, init='k-means++',random_state=1)
cls_shot.fit(X1)
newfeature_shot = cls_shot.labels_ # the labels from kmeans clustering

# append on the second clustering
cls_where = KMeans(n_clusters=n_where, init='k-means++',random_state=1)
cls_where.fit(X2)
newfeature_where = cls_where.labels_ # the labels from kmeans clustering

y = kobe_df5['shot_made_flag']
X = kobe_df5[['period', 'shot_zone_id']]
X = np.column_stack((X, pd.get_dummies(newfeature_shot),pd.get_dummies(n
ewfeature_where)))

cv = StratifiedKFold(y, n_folds=10)
acc = cross_val_score(clf,X,y=y,cv=cv)

clf.fit(X, y)

print "Clusters",n_shot,n_where,"Average accuracy = ", acc.mean()*100,
"+-", acc.std()*100
```

```
Clusters 2 4 Average accuracy =  60.6444806887 +- 3.55802032875
```

Prediction

In [56]:
```
# Shot Location
test_X1 = kobe_df6_pre[['loc_x','loc_y', 'shot_distance']]
newfeature_shot = cls_shot.predict(test_X1)

# Where Factor
test_X2 = kobe_df6_pre[['season_id','game_event_id','action_type_id']]
newfeature_where = cls_where.predict(test_X2)

test_X = kobe_df6_pre[['period', 'shot_zone_id', 'shot_id']]

predictions = clf.predict_proba(np.column_stack((test_X.drop('shot_id',
1), pd.get_dummies(newfeature_shot),pd.get_dummies(newfeature_where))))
submission_6 = pd.DataFrame({'shot_id': test_X.shot_id, 'shot_made_fla
g': predictions[:, 1]})

sub = [submission, submission_6]
submission = pd.concat(sub)

submission_6.head(10)
```

Out[56]:

	shot_id	shot_made_flag
21078	21079	0.468086
21089	21090	0.390970
21092	21093	0.468086
21103	21104	0.359240
21113	21114	0.402919
21116	21117	0.390970
21119	21120	0.210513
21120	21121	0.300062
21131	21132	0.256293
21135	21136	0.386410

Kaggle Submission

In [58]:
```
submission.sort_values(['shot_id'], ascending=[True], inplace=True)
submission.to_csv('submission.csv')
```

We submitted our CSV file from above to the Kaggle for scoring. Our attempt at beating our benchmark received a score of 0.79222. By using K-Means and Random Forest, we were able to decrease our Log-Loss value by 0.24980.

Deployment

The deployment of our algorithm does not necessarily make sense to be used for anything other than shot prediction for Kobe Bryant's career. Each model was created specifically for Kobe Bryant, factoring in coaching changes, a major injury and playing style changes. The only other application is for players on the team (except for the last model, which was created to mark the post injury update). Our goals for our model are to obtain a predictive accuracy that is above random selection. Shooting percentage varies based on location of the shot, so beating random selection is the most pertinent goal. Unfortunately we were unable to obtain or goal. With so many similar shots, and often the same shot situation having different outcomes, a model predicting better than random chance was not accomplished and is lacking utility.

If ideal, our model most likely would be deployed into a gambling, or for player/situation analytics. As for gambling an optimized model will allow for accurate gambling models to produce for a shot by shot basis. A chance metric can be used by teams to design plays and adjust for situations while attempting to maximize a player's chance of making a shot. A this metric can also be used to retroactively analyze decision-making on a shot by shot basis.

Other data that could be collected would be total game minutes (before the shot), continuous stretch of play up to the shot, distance from defender when taking the shot, and the time since most recent game (in days). Each one of these marks speaks to situations regarding fatigue and shot alteration, both can drastically change the outcome of the shot.

An ideal model would be updated within the game after each shot, which can be achieved through dynamic, efficient coding and high amount of computing power for speed. Updating the model after each game would probably the most realistic option.

An optimized model would be fully automated and update on a player by player basis reacting to key changes similar to how Microsoft attempts to predict the outcomes of games based on game and existential factors. Key features could be injuries, coaching changes, private (non-basketball related disputes) and even more. By using social media and Internet driven metrics an algorithm can constantly update to give up to the minute predictions. If Kobe were to break his finger, this type of algorithm could learn this information due to a spike in tweets with the words "Kobe", "Bryant", "Breaks" and "Finger" and alter the outcome of shots taken based on his expected recovery and return date. An ideal model producing algorithm is current beyond our ability, but could be achieved with the proper techniques and computing capabilities.

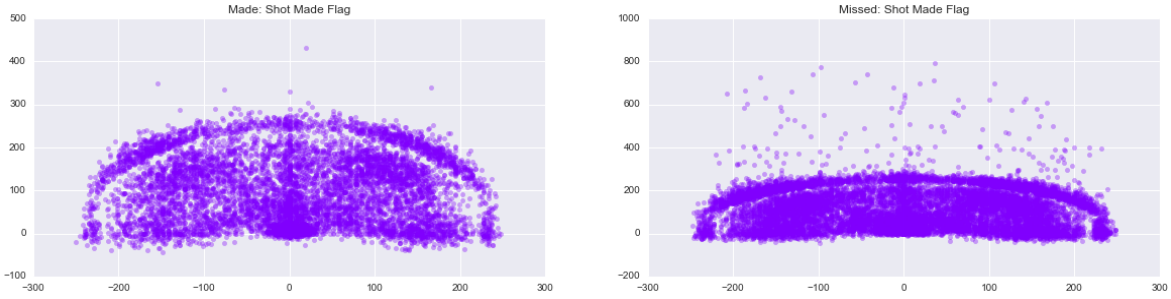
Going Further

Records with 'loc_y' > 300

In a further investigation, we plan to explore the effects of removing the records where the column 'loc_y' is greater than 300. Looking at the plots below, Career Makes vs. Career Misses, we notice that most of his shots above 'loc_y' = 300 are misses. After doing some analysis, Kobe only took about 160 out of about 30000 shots from beyond this point. This means that less than one percent of his shots actually came from beyond the loc_y > 300 mark.

In [17]:
```
plt.figure(figsize=(20,10))

h1 = df.loc[df.shot_made_flag == 1]
h2 = df.loc[df.shot_made_flag == 0]

# Made
plt.subplot(221)
scatter_plot_by_category(h1, 'shot_made_flag')
plt.title('Made: Shot Made Flag')

# Miss
plt.subplot(222)
scatter_plot_by_category(h2, 'shot_made_flag')
plt.title('Missed: Shot Made Flag')
```

Out[17]:
```
<matplotlib.text.Text at 0x13cb79050>
```



In [19]:
```
print df.loc[df.loc_y > 300].shape
```

```
(160, 19)
```

DBSCAN

Instead of using K-Means, we would like to furhter investigate our data set using DBSCAN. We attempted to use DBSCAN, but we were not able to get it running with Random Forests. Below, we have a model based on Kobe's Rookie year. We plan to use this same methodology towards the other models. Hopefully this technique will allow us to decrease our log-loss value, which means we will increase our accuracy.

In [28]:
```
#features_data = kobe_df0[['loc_x', 'loc_y', 'period', 'playoffs', 'shot
_distance', 'time_remaining_sec', 'home_game', 'two_pt_fg', 'season_id',
'action_type_id', 'shot_area_id','shot_made_flag']]
from sklearn.cluster import DBSCAN

clf = RandomForestClassifier(n_estimators=70,random_state=1)

# Shot Location
X1 = kobe_df0[['loc_x','loc_y', 'shot_distance']]
# Where Factor
X2 = kobe_df0[['season_id','game_event_id','action_type_id']]

params = []
for eps in [0.1, 0.125, 0.15]:
    for mpts in range(10, 30, 5):
        # get the first clustering
        cls_shot = DBSCAN(eps=eps, min_samples=mpts)
        cls_shot.fit(X1)
        newfeature_shot = cls_shot.labels_ # the labels from kmeans clus
tering

        # append on the second clustering
        cls_where = DBSCAN(eps=eps, min_samples=mpts)
        cls_where.fit(X2)
        newfeature_where = cls_where.labels_ # the labels from kmeans cl
ustering

        y = kobe_df0['shot_made_flag']
        X = kobe_df0[['home_game', 'time_remaining_sec']]
        X = np.column_stack((X, pd.get_dummies(newfeature_shot),pd.get_d
ummies(newfeature_where)))
        cv = StratifiedKFold(y, n_folds=10)

        acc = cross_val_score(clf,X,y=y,cv=cv)
        params.append((eps,mpts,acc.mean()*100,acc.std()*100)) # save st
ate

        print eps,mpts, acc.mean()*100, "+-", acc.std()*100
```

```
0.1 10 45.7206477733 +- 5.5906240367
0.1 15 45.7206477733 +- 5.5906240367
0.1 20 45.7206477733 +- 5.5906240367
0.1 25 45.7206477733 +- 5.5906240367
0.125 10 45.7206477733 +- 5.5906240367
0.125 15 45.7206477733 +- 5.5906240367
0.125 20 45.7206477733 +- 5.5906240367
0.125 25 45.7206477733 +- 5.5906240367
0.15 10 45.7206477733 +- 5.5906240367
0.15 15 45.7206477733 +- 5.5906240367
0.15 20 45.7206477733 +- 5.5906240367
0.15 25 45.7206477733 +- 5.5906240367
```

In [29]:
```
clf = RandomForestClassifier(n_jobs=-1, n_estimators=70)

eps=0.15
mpts=25

# Shot Location
X1 = kobe_df0[['loc_x','loc_y', 'shot_distance']]
# Where Factor
X2 = kobe_df0[['season_id','game_event_id','action_type_id']]

dbscan_test = kobe_df0

cls_shot = DBSCAN(eps=eps, min_samples=mpts)
cls_shot.fit(X1)
dbscan_test['newfeature_shot'] = cls_shot.labels_

cls_where = DBSCAN(eps=eps, min_samples=mpts)
cls_where.fit(X2)
dbscan_test['newfeature_where'] = cls_where.labels_

dbscan_test.head(10)
```

```
/Library/Python/2.7/site-packages/ipykernel/__main__.py:15: SettingWith
CopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-d
ocs/stable/indexing.html#indexing-view-versus-copy
/Library/Python/2.7/site-packages/ipykernel/__main__.py:19: SettingWith
CopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-d
ocs/stable/indexing.html#indexing-view-versus-copy
```

Out[29]:

	game_event_id	loc_x	loc_y	period	playoffs	shot_distance	shot_made
22901	102	-140	116	1	0	18	0
22902	127	-131	97	2	0	16	0
22903	124	-142	181	2	0	23	1
22904	144	0	0	2	0	0	0
22905	151	-10	138	2	0	13	1
22907	226	-64	223	2	0	23	1
22909	334	-79	177	3	0	19	0
22910	337	-103	207	3	0	23	1
22911	352	0	0	3	0	0	0
22912	380	-155	175	4	0	23	0

10 rows × 22 columns

In [31]:
```
plt.figure(figsize=(20,10))

h1 = dbscan_test.loc[dbscan_test.shot_made_flag == 1]
h2 = dbscan_test.loc[dbscan_test.shot_made_flag == 0]

# Made
plt.subplot(221)
scatter_plot_by_category(h1, 'newfeature_shot')
plt.title('Made: newfeature_shot')

plt.subplot(222)
scatter_plot_by_category(h1, 'newfeature_where')
plt.title('Made: newfeature_where')

# Made
plt.subplot(223)
scatter_plot_by_category(h2, 'newfeature_shot')
plt.title('Miss: newfeature_shot')

plt.subplot(224)
scatter_plot_by_category(h2, 'newfeature_where')
plt.title('Miss: newfeature_where')
```

```
Out[31]: <matplotlib.text.Text at 0x1182d5ed0>
```

Sources:

https://github.com/eclarson/DataMiningNotebooks/blob/master/09.%20Clustering%20and%20Discretizati
(https://github.com/eclarson/DataMiningNotebooks/blob/master/09.%20Clustering%20and%20Discretizat

https://www.kaggle.com/arjoonn/kobe-bryant-shot-selection/preliminary-exploration
(https://www.kaggle.com/arjoonn/kobe-bryant-shot-selection/preliminary-exploration)

https://www.kaggle.com/dixhom/kobe-bryant-shot-selection/data-analysis-for-beginners
(https://www.kaggle.com/dixhom/kobe-bryant-shot-selection/data-analysis-for-beginners)

http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
(http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html)

http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html (http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html)

http://www.bogotobogo.com/python/python_numpy_array_tutorial_basic_A.php
(http://www.bogotobogo.com/python/python_numpy_array_tutorial_basic_A.php)