Part 1: Project Objective

NCAA Division I Men's Basketball Tournament currently featuring 68 college basketball teams from 350 teams.

Our objective: predict and simulate Final Results at 2018 NCAA Basketball Tournament

Part 2: Description of Dataset

2017 & 2018 NCAA Regular Season Champion data including team and player statistics

For Team dataset:

* Each row represents data for a single game
* First half columns represent Home Team's performance
* Second half columns represent Away Team's performance
  e.g. PTS(Total Points), 3P% (3-Point Field Goal Percentage)
* After cleaning, we have 351 teams and 10795 games

In [59]:
```
df1.head(0)
```

Out[59]:

	Unnamed: 0	Team	MP	FG	FGA	FG%	2P	2PA	2P%	3P	3PA	3P%	FT	FTA	FT%	O

In [60]:
```
#### Maryland 2017 & 2018 winning rate in regular season!
100 * len(df1[(df1.Team == 'maryland')][(df1['Win?'] == 1)])/len(df1[(
df1.Team == 'maryland')])
```

Out[60]: 64.51612903225806

For Player dataset:

* Usually people think key player can carry the whole team, but we find it 's not true!
* We identified key players who are ranked on top of a single metric(E.g. points per game or 3P%)
* BUT these player's team even cannot get into the final rounds

In [12]:
```
# show top 10 players and their teams using mean of highest points per game (PTS) for each player
top_player=df2[['player','Team','PTS']].groupby(['player','Team']).mean().sort_values(['PTS'], ascending=False).iloc[0:10]
top_player
```

Out[12]:

		PTS
player	Team	
DJ Hanes	nebraska-christian	39.00000
Jalen Adams	olivet	37.00000
Taylor Gilpin	johnson-university	32.00000
Keith Hayes	william-jewell	31.00000
George Brock	southern-new-orleans	31.00000
Denzel Famble	voorhees	30.00000
Brian Cameron	wesley	30.00000
Marcus Keene	central-michigan	29.96875
Rob Davis	fort-hays-state	29.00000
Tracy Edmond	olivet	29.00000

Part 3: Feature Engineering

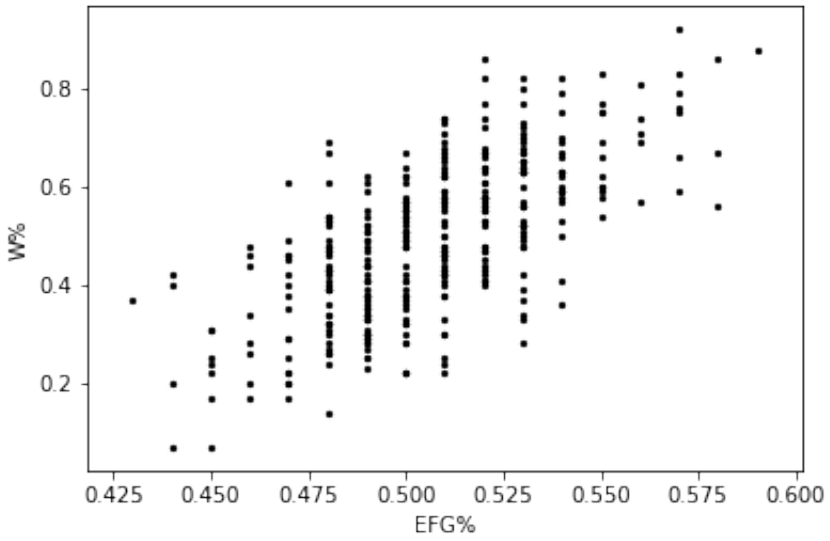
```
* Average performance data for each team between 2017-2018 based on existi
ng data;
* Add 23 new features based on current and outsource dataset;
* E.g. Margin of Victory = PTS - Opp PTS;
* E.g. True Shooting Percentage is shooting efficiency uses field goals, 3
-point field goals, and free throws.
```

In [40]:
```
# top 5 wining rate teams during 2017-2018 and their statistics
teamStats.sort_values(['W%'], ascending=False).head(5)
```

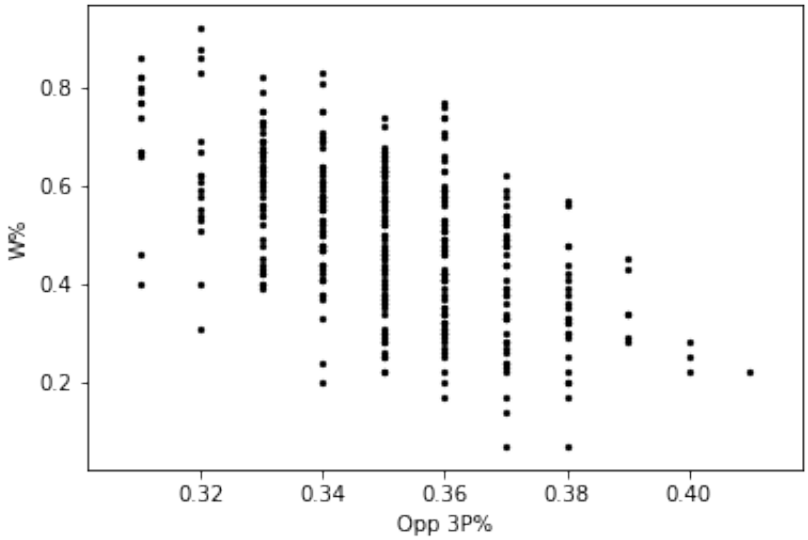
Out[40]:

	MP	FG	FGA	FG%	2P	2PA	2P%	3P	3PA	3P%	FT	FT
gonzaga	201.37	30.01	59.51	0.51	21.89	37.97	0.58	8.13	21.54	0.38	15.23	21.1
villanova	201.09	28.88	57.58	0.50	18.86	31.70	0.60	10.03	25.88	0.38	14.29	18.2
cincinnati	200.72	26.39	58.00	0.45	19.16	37.42	0.51	7.23	20.58	0.35	13.88	20.1
saint-marys-ca	200.76	27.20	53.68	0.51	18.97	33.15	0.57	8.23	20.53	0.40	11.98	15.7
arizona	201.45	27.41	56.14	0.49	20.67	38.52	0.54	6.74	17.62	0.38	16.72	21.9

In [42]:
```
# Effective Field Goal Percentage (eFG%) with wining rate visualizatio
n
teamStats.plot(kind='scatter', x='EFG%', y='W%', color='k', marker='.'
);
```



In [43]:
```
# Opp 3P (Opponents 3 Point Field Goals Made) with winging rate
teamStats.plot(kind='scatter', x='Opp 3P%', y='W%', color='k', marker=
'.');
```



Part 4: Feature Transformation

```
* Our X: For each game, the features' differences between two teams' avera
ge statistics
* Our Y: whether will win or not (0 or 1)
```

In [54]:
```
def compareTwoTeams(id_1, id_2):
    team_1 = getSeasonData(id_1)
    team_2 = getSeasonData(id_2)
    diff = [float(a) - float(b) for a, b in zip(team_1, team_2)]
    diff.pop(0)
    return diff
```

Part 5: Modeling Preperation

- We decide do not do normalization process, because our data is relatively evenly distributed.
- Create Training, Validation and Testing Dataset

In [65]:
```
# Full dataset after feature transformation
x.head(5)
```

Out[65]:

	MP	FG	FGA	FG%	2P	2PA	2P%	3P	3PA	3P%	FT	FTA	FT%	OR
0	-0.83	-1.36	-3.08	0.00	-1.89	-4.02	0.01	0.53	0.93	0.01	-1.57	-3.14	0.05	-1.6
1	0.83	1.36	3.08	0.00	1.89	4.02	-0.01	-0.53	-0.93	-0.01	1.57	3.14	-0.05	1.6
2	0.66	-0.39	-2.27	0.01	-0.02	-0.73	0.01	-0.37	-1.54	0.01	-0.72	-0.12	-0.02	-0.3
3	-0.66	0.39	2.27	-0.01	0.02	0.73	-0.01	0.37	1.54	-0.01	0.72	0.12	0.02	0.3
4	1.84	-4.53	-2.21	-0.06	-6.56	-8.14	-0.07	2.03	5.94	-0.01	-2.06	-1.77	-0.06	-0.8

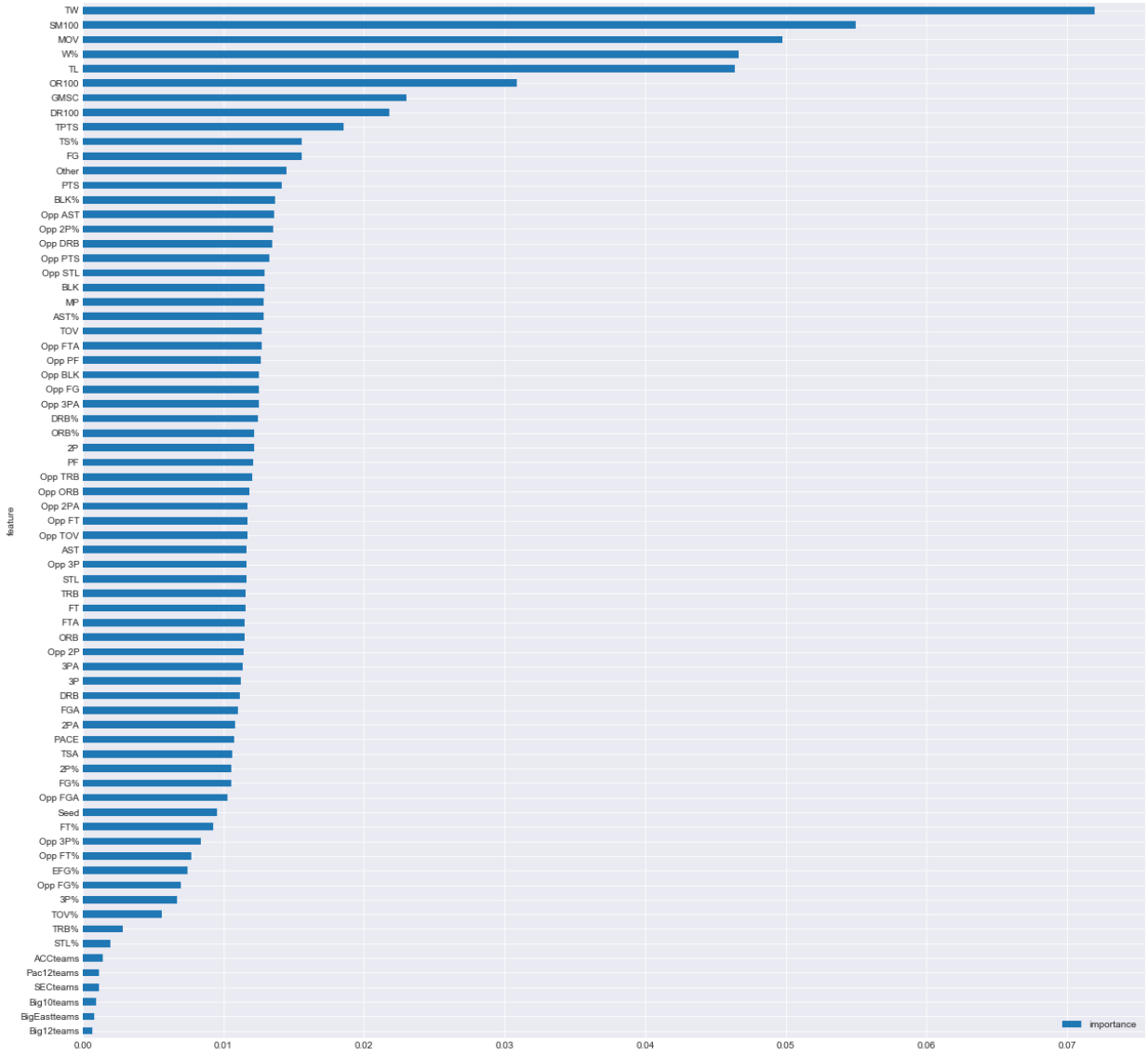
Part 6: Modeling

- Check general features significance based on random forest
  1. Using grid search to find the best parameter;
  2. Performing stepwise feature selections for models;
  3. Performing cross validation on validation dataset to select best models;
  4. Testing accuarcy on Testing Dataset;
  5. Using voting strategy assembling 4 models and Neural Network Model;

Parameter Insights Visualization

In [68]:
```
sns.set_style('darkgrid')
features = pd.DataFrame()
features['feature'] = x.columns
features['importance'] = model2.feature_importances_
features.sort_values(by=['importance'], ascending=True, inplace=True)
features.set_index('feature', inplace=True)
features.plot(kind='barh', figsize=(20, 20))
```

Out[68]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a330fb400>
```

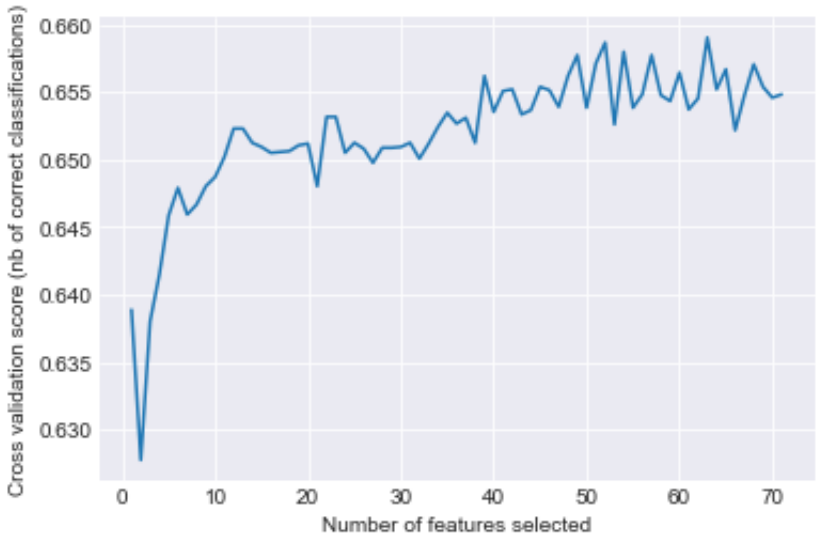


Feature Selection Example and Visualization

In [69]:
```
rfecv = RFECV(estimator=model2, step=1, cv=StratifiedKFold(3),
              scoring='accuracy')
rfecv.fit(X_prep,Y_prep)
print("Optimal number of features : %d" % rfecv.n_features_)

# Plot number of features VS. cross-validation scores
plt.figure()
sns.set_style('darkgrid')
plt.xlabel("Number of features selected")
plt.ylabel("Cross validation score (nb of correct classifications)")
plt.plot(range(1, len(rfecv.grid_scores_) + 1), rfecv.grid_scores_)
plt.show()
```

```
Optimal number of features : 63
```



Grid serach example

In [81]:
```
# logistic regression
model1 = linear_model.LogisticRegression()
model1_params = {'penalty':['l1','l2']}
model1_grid = GridSearchCV(estimator=model1, param_grid=model1_params,
scoring='accuracy', cv=5, n_jobs=4)
%time model1_grid.fit(X_prep, Y_prep)
```

```
CPU times: user 2.47 s, sys: 287 ms, total: 2.76 s
Wall time: 15.4 s
```

Out[81]:
```
GridSearchCV(cv=5, error_score='raise',
       estimator=LogisticRegression(C=1.0, class_weight=None, dual=F
alse, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jo
bs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0
.0001,
          verbose=0, warm_start=False),
       fit_params=None, iid=True, n_jobs=4,
       param_grid={'penalty': ['l1', 'l2']}, pre_dispatch='2*n_jobs'
,
       refit=True, return_train_score='warn', scoring='accuracy',
       verbose=0)
```

In [82]:
```
print(model1_grid.best_score_)
print(model1_grid.best_params_)
```

```
0.7196763833992095
{'penalty': 'l1'}
```

Neural Network Example

In [126]:
```
labels = to_categorical(np.asarray(Y_prep))
adam=Adam()
# Assembling MLP
model = Sequential()
model.add(Dense(48, input_shape=(71,), activation='relu'))
model.add(Dropout(0.05))
model.add(Dense(labels.shape[1], activation='sigmoid'))
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_29 (Dense)             (None, 48)                3456      
_________________________________________________________________
dropout_15 (Dropout)         (None, 48)                0         
_________________________________________________________________
dense_30 (Dense)             (None, 2)                 98        
=================================================================
Total params: 3,554
Trainable params: 3,554
Non-trainable params: 0
_________________________________________________________________
```

Voting Strategy Example

In [128]:
```
def predictTwoTeams(team1_name,team2_name):
    team1_id = teamIndex[teamIndex['Team']==team1_name].values[0][0]
    team2_id = teamIndex[teamIndex['Team']==team2_name].values[0][0]

    ## voting
    predResult1 = model1.predict([compareTwoTeams(team1_id,team2_id)])
    predResult3 = model3.predict([compareTwoTeams(team1_id,team2_id)])
    predResult6 = model6.predict([compareTwoTeams(team1_id,team2_id)])
    predResult10 = model10.predict([compareTwoTeams(team1_id,team2_id)
])
    predResult_n = model.predict(np.array([compareTwoTeams(team1_id,te
am2_id)]))

    if (predResult1[0]+predResult3[0]+predResult6[0]+predResult10[0]+p
redResult_n[0][1])/5 >= 0.5:
        return team1_name
    else:
        return team2_name
```

Part 7: Simulate Tournament

In [129]:
```
simulate_tournament()
```

```
First Round: 64 >>> 32
south
virginia VS maryland-baltimore-county : virginia win!!!
creighton VS kansas-state : kansas-state win!!!
kentucky VS davidson : kentucky win!!!
arizona VS buffalo : arizona win!!!
miami-fl VS loyola-il : miami-fl win!!!
tennessee VS wright-state : tennessee win!!!
nevada VS texas : nevada win!!!
cincinnati VS georgia-state : cincinnati win!!!


west
xavier VS texas-southern : xavier win!!!
missouri VS florida-state : florida-state win!!!
ohio-state VS south-dakota-state : ohio-state win!!!
gonzaga VS north-carolina-greensboro : gonzaga win!!!
houston VS san-diego-state : houston win!!!
michigan VS montana : michigan win!!!
texas-am VS providence : providence win!!!
north-carolina-state VS lipscomb : north-carolina-state win!!!


east
villanova VS radford : villanova win!!!
virginia-tech VS alabama : virginia-tech win!!!
west-virginia VS murray-state : west-virginia win!!!
wichita-state VS marshall : wichita-state win!!!
florida VS st-bonaventure : florida win!!!
texas-tech VS stephen-f-austin : texas-tech win!!!
arkansas VS butler : arkansas win!!!
purdue VS cal-state-fullerton : purdue win!!!


midwest
kansas VS pennsylvania : kansas win!!!
seton-hall VS north-carolina-state : seton-hall win!!!
clemson VS new-mexico-state : clemson win!!!
auburn VS college-of-charleston : auburn win!!!
texas-christian VS syracuse : texas-christian win!!!
michigan-state VS bucknell : michigan-state win!!!
rhode-island VS oklahoma : rhode-island win!!!
duke VS iona : duke win!!!
```

```
second round 32>>>16
virginia VS kansas-state : virginia win!!!
kentucky VS arizona : arizona win!!!
miami-fl VS tennessee : tennessee win!!!
nevada VS cincinnati : cincinnati win!!!
xavier VS florida-state : xavier win!!!
ohio-state VS gonzaga : gonzaga win!!!
houston VS michigan : michigan win!!!
providence VS north-carolina-state : providence win!!!
villanova VS virginia-tech : villanova win!!!
west-virginia VS wichita-state : west-virginia win!!!
florida VS texas-tech : florida win!!!
arkansas VS purdue : purdue win!!!
kansas VS seton-hall : kansas win!!!
clemson VS auburn : auburn win!!!
texas-christian VS michigan-state : michigan-state win!!!
rhode-island VS duke : duke win!!!



third round 16>>>8
virginia VS arizona : virginia win!!!
tennessee VS cincinnati : cincinnati win!!!
xavier VS gonzaga : gonzaga win!!!
michigan VS providence : michigan win!!!
villanova VS west-virginia : villanova win!!!
florida VS purdue : purdue win!!!
kansas VS auburn : kansas win!!!
michigan-state VS duke : duke win!!!


final four 8>>>4
virginia VS cincinnati : virginia win!!!
gonzaga VS michigan : michigan win!!!
villanova VS purdue : villanova win!!!
kansas VS duke : kansas win!!!



semifinals 4>>>2
virginia VS michigan : virginia win!!!
villanova VS kansas : kansas win!!!


Championship!!!!!
Congratulation!!!! kansas
```

Our overall accuracy is around 70% compared to this year's results