follow the Best of 3. I selected the three classifiers as Logistic Regression, Decision Trees and Random Forest to make the predictions on both structures. Plotting the ROC curves along with win probability gives us a fair idea for predicting.

3. Implementation

3.1. Data Set

The dataset that I am using can be obtained from http://www.tennis-data.co.uk/data.php or downloaded from UCI ATP repository.

	ATP	Location	Tournament	Date	Series	Court	Surface	Round	Best of	Winner	...	Wsets	Lsets	Comment	PSW	PSL	B365W	B365L
0	1	Adelaide	Australian Hardcourt Championships	2000-01-03	International	Outdoor	Hard	1st Round	3	Dosedel S.	...	2.0	0.0	Completed	NaN	NaN	NaN	NaN
1	3	Doha	Qatar Open	2000-01-03	International	Outdoor	Hard	1st Round	3	Kiefer N.	...	2.0	0.0	Completed	NaN	NaN	NaN	NaN
2	3	Doha	Qatar Open	2000-01-03	International	Outdoor	Hard	1st Round	3	Gaudio G.	...	2.0	1.0	Completed	NaN	NaN	NaN	NaN
3	3	Doha	Qatar Open	2000-01-03	International	Outdoor	Hard	1st Round	3	El Aynaoui Y.	...	2.0	1.0	Completed	NaN	NaN	NaN	NaN
4	3	Doha	Qatar Open	2000-01-03	International	Outdoor	Hard	1st Round	3	Cherkasov A.	...	2.0	0.0	Completed	NaN	NaN	NaN	NaN

5 rows × 23 columns

Figure 2. The original dataset is shown above

The dataset is varied and has data ranging from float to categorical. Some of the ranges of the features are:

- Dates: 2000 – 2018
- Surface: {Hard, Clay, Grass, Carpet}
- Series: {International, Grand Slam, Masters, ATP250, ATP500, Masters-1000, International Gold}
- Comment: {Completed, Retired}
- Round: {1st Round, 2nd Round, 3rd Round, 4th Round, Quarterfinals, Semifinals, The Finals}
- Court: {Indoor, Outdoor}

Out of all these 23 features, I selected only 7 features which majorly seem to affect the prediction of winner.

3.2. Preprocessing, Feature Extraction, Dimensionality Adjustment

A primary test run of the classifiers Logistic Regression, Decision Trees and Random Forest on the complete data set gave me the accuracies on test data as:

Dataset	Logistic Regression	Decision Trees	Random Forests
Complete	0.6259	0.5788	0.5793

Table 1

So out of all the features, the features I selected are:

i) Date: date of the match

ii) Series: name of ATP tennis series (we kept the four main current categories namely Grand Slams, Masters 1000, ATP250, ATP500)

iii) Surface: type of surface (clay, hard or grass)

iv) Round: round of match (from first round to the final)

v) Best of: maximum number of sets playable in match (Best of 3 or Best of 5)

vi) WRank: ATP Entry ranking of the match winner as of the start of the tournament

vii) LRank: ATP Entry ranking of the match loser as of the start of the tournament

A basic run of the logistic regression, Decision tree and Random forest classifier on the whole data with selected features gave me accuracies of:

Dataset	Logistic Regression	Decision Trees	Random Forest
Selected features spanning over all the years from 2000 - 2018	0.66	0.58	0.59

Table 2

- Now proceeding with final feature selection, I kept only completed matches i.e. eliminated matches with injury withdrawals and walkovers.
- For convenience renamed Best of to Best_of.
- Dropped all the NaN entries, errors in the dataset and unimportant entries (e.g. there are very few entries for master's Cup)
- Considered the two final years only (to avoid comparing different categories of tournaments which existed in the past). Note that this choice is somewhat arbitrary and can be changed if needed.
- Chose only higher ranked players for better accuracy
- Transform strings('Best_of','WRank','LRank') into numerical values

Further, added a new column named ‘win’ for the variable win described above using an auxiliary function win(x). The output variable is binary. The better player has higher rank. The design of the win function:

win=1, if higher ranked player wins

win=0, if higher ranked player loses

A previous analysis by some researchers as Corral and Prieto-Rodriguez [2], we are restricting our analysis to top players as including everyone decreases the predictive power.

	Date	Series	Surface	Round	Best_of	WRank	LRank	win
36209	2015-01-05	ATP250	Hard	1st Round	3	52	40	0
36211	2015-01-05	ATP250	Hard	1st Round	3	26	41	1
36212	2015-01-05	ATP250	Hard	1st Round	3	98	63	0
36213	2015-01-05	ATP250	Hard	1st Round	3	33	66	1
36214	2015-01-05	ATP250	Hard	1st Round	3	27	30	1

Figure 3. Our new processed dataset

Restricting our analysis to matches of Best_of = 5. Since only Grand Slams have 5 sets, we can drop the new Series column. The case of Best_of = 3 will be considered later.

Now we notice that our dataset is uneven in terms of frequency of wins, which presents a case of imbalance.

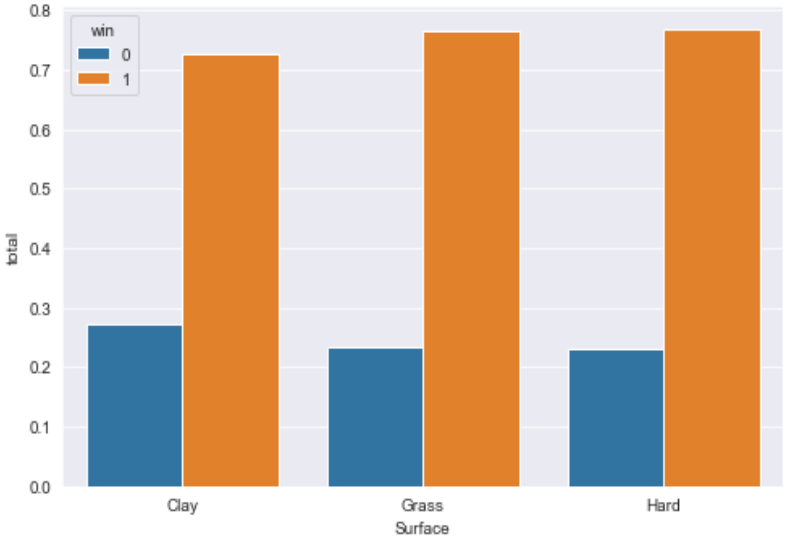

Figure 4. Looking at the imbalance ‘surface’ wise

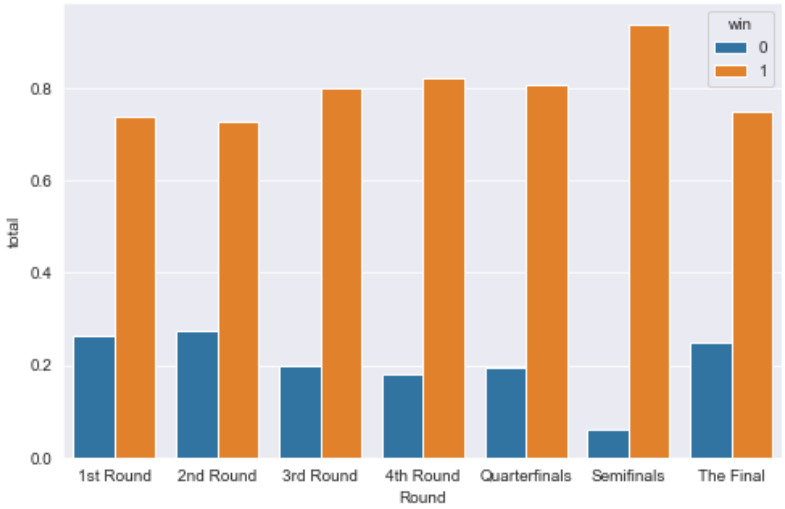

Figure 5. Looking at the imbalance 'rounds' wise

Dataset is uneven in terms of frequency of wins, balance this using stratified sampling procedure.

Exploratory Analysis for Best_of = 5

We first look at percentage of wins for each surface. We find that when the Surface is Clay there is a higher likelihood of upsets (opposite of wins) i.e. the percentage of wins is lower. The difference is not too large tough.

After Stratification Results:

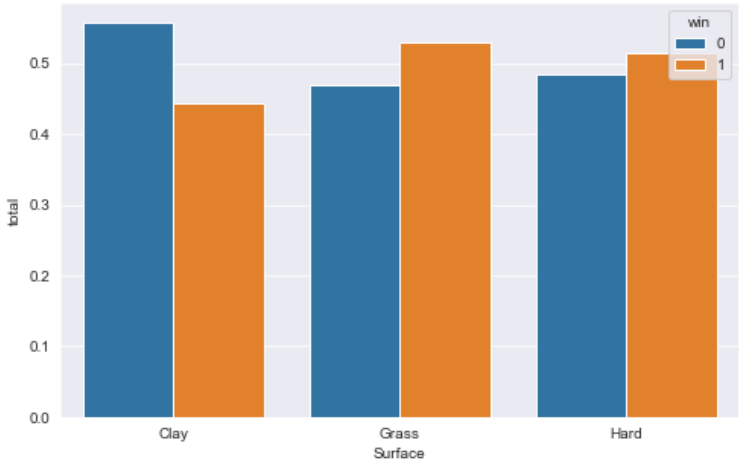

Figure 6. After stratification, wins by 'surface'

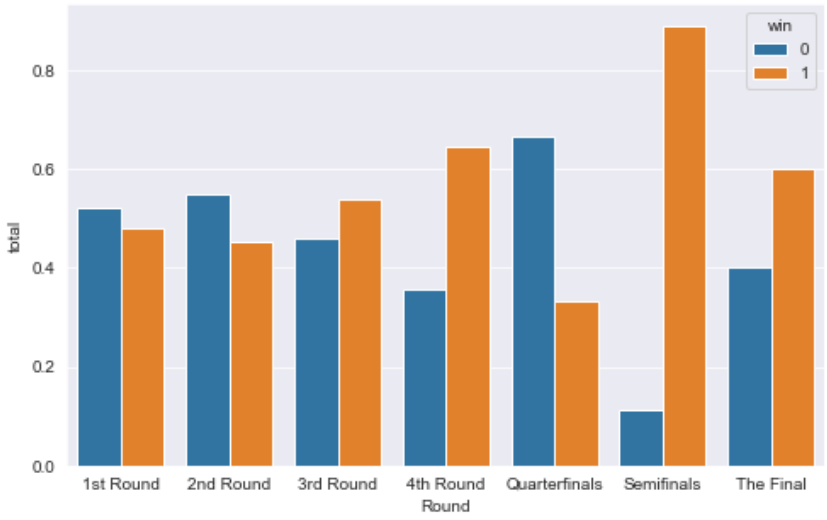

Figure 7. After stratification, wins by 'rounds'

A slight trend of higher ranked player winning emerges from semifinals and finals.

To facilitate coding, then converted the rounds and surface data to categorical data

3.3. Dataset Methodology

Creating the "D" column:

This is perhaps the innovative idea which should be explored more. We create a new column "D" for difference between the logarithmic values of the ranks between the two give players. This is a trump card feature as similar ranking people will have no difference in play, esp. when the ranks are extremely low. Difference between the **logarithms** of the rankings defined by **D** is to consider the non-linearity of the players quality (e.g. for bottom ranked players a difference of one position corresponds to effectively no quality difference)

While cutting down on all the columns and redundant data has left me with little data to work with, these features however present the best accuracy as discussed in the later stages. The total dataset split into training and testing, of 1392 and 464 respectively. We keep all the other 9 features with it.

Also going ahead, I used cross validation in all the three classifiers, with k = 5 (default and externally specified as well). The test set was finally used to get the accuracy and I made sure that the test set doesn't get tampered with till the last prediction. So, performing a k-fold cross validation to build the model and test the model on the test data set is my preferred choice of execution. Cross validation is in general a reasonably accurate estimate of out-of-sample accuracy

3.4. Training Process

Model 1: Logistic Regression

Defining the logit function:

```
def logit(x):
    return np.exp(x) / (1 + np.exp(x))
```

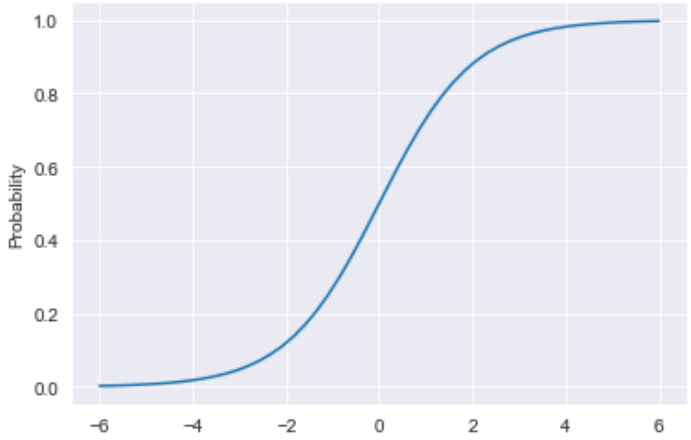

Figure 8. Logit function on probability

Logit Regression Results

Dep. Variable:	win	No. Observations:	397
Model:	Logit	Df Residuals:	387
Method:	MLE	Df Model:	9
Date:	Sat, 07 Dec 2019	Pseudo R-squ.:	0.1329
Time:	17:50:51	Log-Likelihood:	-238.61
converged:	True	LL-Null:	-275.18
Covariance Type:	nonrobust	LLR p-value:	3.679e-12

	coef	std err	z	P>|z|	[0.025	0.975]
const	-1.2314	0.279	-4.419	0.000	-1.778	-0.685
Round_2	-0.4406	0.269	-1.639	0.101	-0.967	0.086
Round_3	-0.1713	0.353	-0.485	0.628	-0.864	0.521
Round_4	0.0981	0.442	0.222	0.824	-0.767	0.964
Round_5	-1.1692	0.798	-1.464	0.143	-2.734	0.396
Round_6	1.8158	1.109	1.637	0.102	-0.358	3.990
Round_7	0.3638	0.954	0.381	0.703	-1.507	2.234
Surface_Grass	0.3491	0.309	1.130	0.258	-0.256	0.954
Surface_Hard	0.2632	0.267	0.987	0.324	-0.260	0.786
D	0.7339	0.108	6.802	0.000	0.522	0.945

Figure 9. Logit Results

Executing the Logistic regression with “lbfgs” solver gives us this ROC curve. Also, to control the said overfitting lasso penalty can be used.

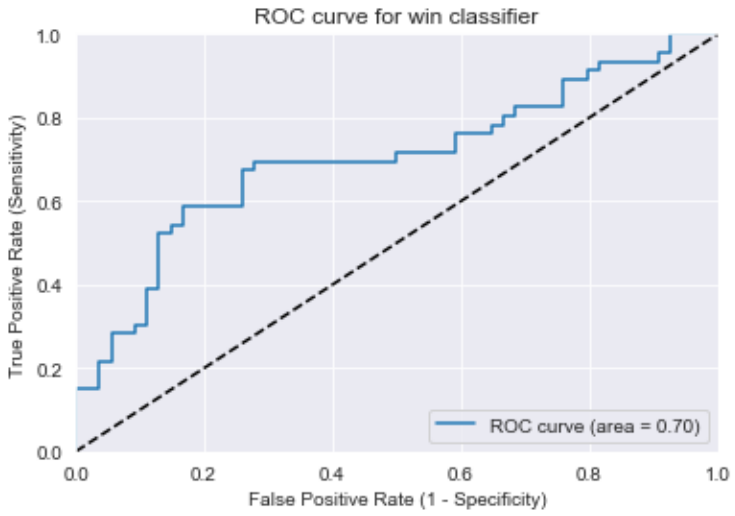

Figure 10. AUC for Best_of = 5

The ROC curve shows how the sensitivity (when the actual value if true how often the prediction is correct) and specificity (when the actual value is negative, how often is the prediction correct) are affected by thresholds (a threshold is used by to convert predicted probabilities into class predictions)

Ideally, the ROC curve should **hug the top left corner** (model predicts upsets and non-upsets more precisely). This can be quantified by the area under the curve AUC. I obtained AUC = 0.62 for 3-sets matches and AUC = 0.75 for 5-sets matches

Win probability dependence on the difference of log-rankings for Best_of = 3 or 5

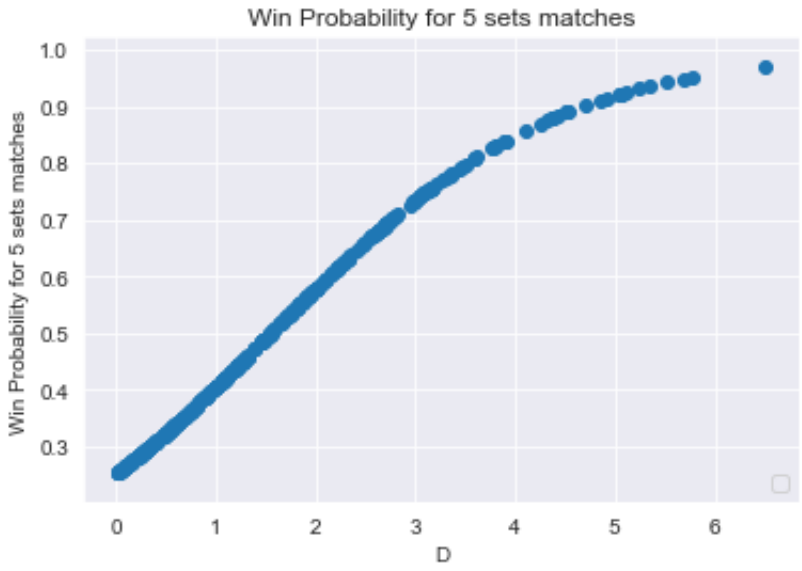

Figure 11. Win Probability

Model 2: Decision Tree

Next, I implemented the decision trees. I chose decision trees as a classifier as I believed a nonlinear classifier can give better results as the features are varied. Also, one of the biggest applications of Decision Tree classically has been in forecasting probabilities which is what we are dealing with here. My parameters are selected heuristically and are:

> DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None, max_features=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, presort=False, random_state=None, splitter='best')

Evaluated the decision tree using cross-validation, using AUC as the evaluation metric. Control for overfitting here is through adjusting the maximum number of questions or minimum number of records in each final node. We use the Gini index for find the variance of classes.

Model 3: Random Forests

The final classifier I worked with was the ensembles of decision trees together, i.e Random Forests.

Random Forests are ensembles of decision trees and have built-in protection against overfitting. Hence for the limited data I have, this becomes a good classifier.

Also, with Random Forests I could extract the importance of the features. This importance of the feature can automatically guide us towards those to be must included features.

Here as you can see, ranking difference, i.e the “D” variable that I created earlier, plays the major role in classification and hence can be said to be the most important feature in our current dataset.

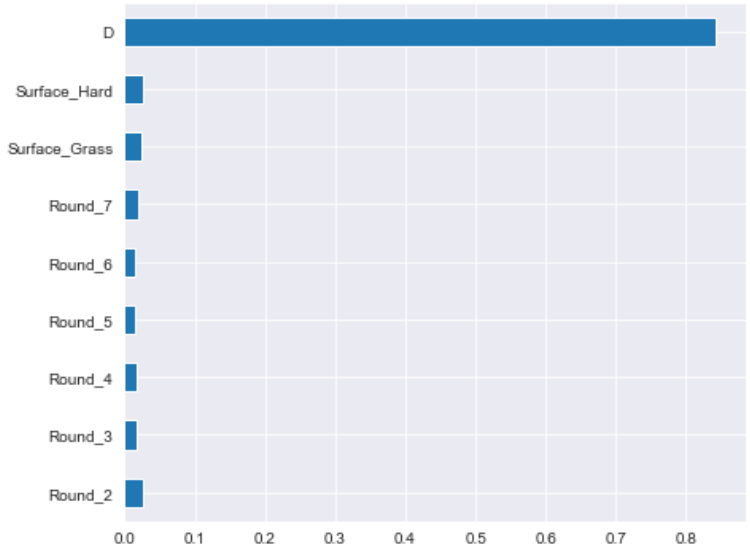

Figure 12. Feature Importance

Parameters here are selected heuristically:

> RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini', max_depth=None, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, n_estimators=200, n_jobs=None, oob_score=False, random_state=None, verbose=0, warm_start=False)

Evaluating the model using cross-validation, I increase the number of estimators to see how that improves the predictive performance.

3.5. Model Selection and Comparison of Results

Till now, I have discussed the performance of the classifiers on the Best_of = 5. Now, let's look at the Best_of = 3 dataset, and the performance of the classifiers on it.

DATA ANALYSIS:

As in the case of 5 sets matches, the frequency of upsets depends weakly on the surface type (data before stratification) The frequency of wins is not strongly dependent on the round and tends to become more uniform at the end of the tournament

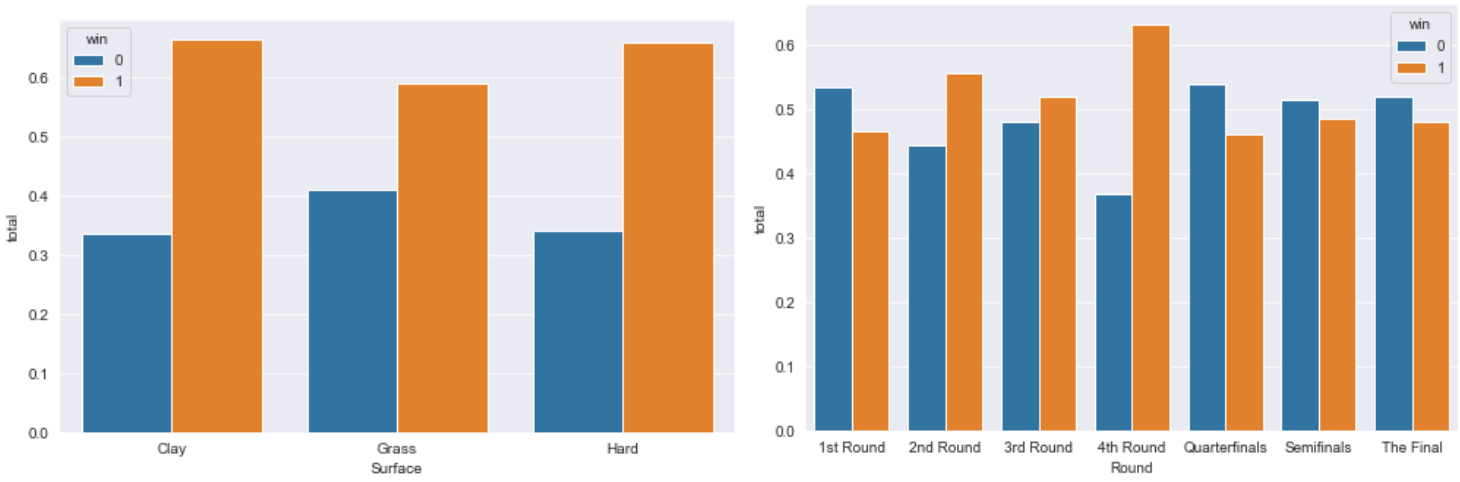

Figure 13. Data before stratification shows the imbalance of wins

Model 1: Logistic Regression

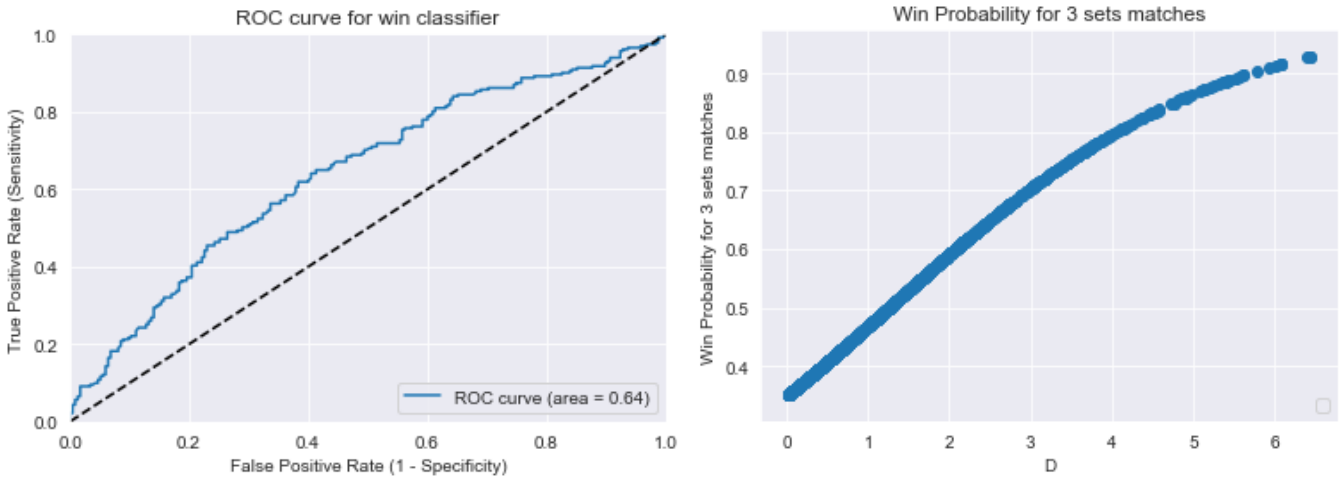

Figure 14. The AUC and the Win probability curves

The AUC decreases as seen in the images for the Best_of = 3 and win prop takes a slightly linear curve. The parameters, cross validation and the model design remains the same.

Model 2: Decision Tree and Model 3: Random Forests

The model design for all the decision tree models remains the same. I am using K = 5 here as well for the cross validation.

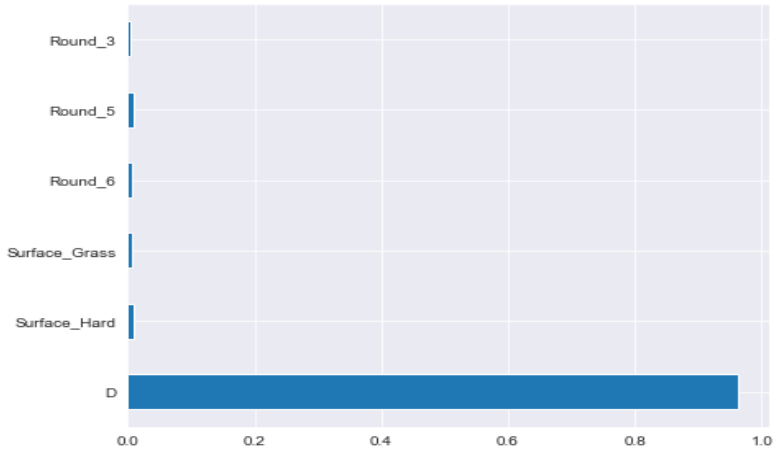

Figure 15. Important features for Best_of = 3

Even for best_of = 3, the defined variable “D” holds the highest importance as seen above.

PERFORMANCE COMPARISION:

1. **LOGISTIC REGRESSION:**

   a) Best_of = 5 performace AUC

```
In [19]: y_pred_class = logreg.predict(X_test)
    ...: from sklearn import metrics
    ...: print(metrics.accuracy_score(y_test, y_pred_class))
    ...: print('True:', y_test.values[0:40])
    ...: print('Pred:', y_pred_class[0:40])
    ...: y_pred_prob = logreg.predict_proba(X_test)[:, 1]
    ...: auc_score = metrics.roc_auc_score(y_test, y_pred_prob)
    ...: auc_score
0.69
True: [0 0 0 0 0 0 1 0 0 1 1 1 0 0 1 0 0 1 0 1 1 1 1 0 1 1 1 1 1 0 1 1 0 1 0 0 0
 1 1 0]
Pred: [0 0 0 1 0 0 1 0 1 1 1 1 0 0 1 0 0 0 0 1 0 1 0 0 1 0 1 1 0 0 1 1 0 1 0 1 0
 0 1 0]
Out[19]: 0.7016908212560387
```

   b) Best_of = 3 performace AUC

```
In [37]: print(metrics.accuracy_score(y_test, y_pred_class))
    ...: y_pred_prob = logreg.predict_proba(X_test)[:, 1]
    ...: auc_score = metrics.roc_auc_score(y_test, y_pred_prob)
    ...: print(auc_score)
    ...: y_pred_prob = logreg.predict_proba(X_test)[:, 1]
    ...: auc_score = metrics.roc_auc_score(y_test, y_pred_prob)
    ...: auc_score
0.6051502145922747
0.6425531914893616
Out[37]: 0.6425531914893616
```

2. **DECISION TREE:**

   a) Best_of = 5 performace AUC

```
In [25]: model = DecisionTreeClassifier(
    ...:                 max_depth = 10,
    ...:                 min_samples_leaf = 8)
    ...: model.fit(X, y)
    ...: filename1 = 'dt.sav'
    ...: pickle.dump(model, open(filename1, 'wb'))
    ...: scores = cross_val_score(model, X, y, scoring='roc_auc', cv=5)
    ...: print('CV AUC {}, Average AUC {}'.format(scores, scores.mean()))
CV AUC [0.6575     0.710625   0.6309375  0.6849359  0.66042078], Average AUC
0.6688838346482577
```

   b) Best_of = 3 performace AUC

```
In [42]: model = DecisionTreeClassifier(
    ...:                 max_depth = 10,
    ...:                 min_samples_leaf = 8)
    ...: model.fit(X, y)
    ...: scores = cross_val_score(model, X, y, scoring='roc_auc', cv=5)
    ...: print('CV AUC {}, Average AUC {}'.format(scores, scores.mean()))
CV AUC [0.60791501 0.63075729 0.57989765 0.57358    0.60036327], Average AUC
0.5985026427151664
```

3. RANDOM FOREST:

a) Best_of = 5 performace AUC

```
In [28]: from sklearn.model_selection import cross_val_score
    ...: scores = cross_val_score(model1, X, y, scoring='roc_auc', cv= 5)
    ...: print('AUC {}, Average AUC {}'.format(scores, scores.mean()))
    ...: for n_trees in range(1, 200, 10):
    ...:     model = RandomForestClassifier(n_estimators = n_trees)
    ...:     scores = cross_val_score(model, X, y, scoring='roc_auc', cv=5)
    ...:     print('n trees: {}, CV AUC {}, Average AUC {}'.format(n_trees, scores,
scores.mean()))
AUC [0.6559375  0.7434375  0.6596875  0.68044872 0.60289283], Average AUC
0.6684808103221564
n trees: 1, CV AUC [0.570625   0.6        0.5875     0.63301282 0.53353057],
Average AUC 0.5849336785009862
n trees: 11, CV AUC [0.6684375  0.6803125  0.6325     0.59423077 0.63116371],
Average AUC 0.6413288954635108
n trees: 21, CV AUC [0.6403125  0.7334375  0.66       0.69198718 0.6183432 ],
Average AUC 0.6688160749506903
n trees: 31, CV AUC [0.6390625  0.7328125  0.64375    0.67467949 0.6226167 ],
Average AUC 0.6625842373438527
n trees: 41, CV AUC [0.635625   0.7475     0.675      0.68269231 0.60190664],
Average AUC 0.6685447896120974
n trees: 51, CV AUC [0.641875   0.743125   0.64625    0.65961538 0.61867193],
Average AUC 0.6619074621959238
n trees: 61, CV AUC [0.6628125  0.7628125  0.6434375  0.66442308 0.61012492],
```

b) Best_of = 3 performace AUC

```
In [46]: from sklearn.model_selection import cross_val_score
    ...: scores = cross_val_score(model, X, y, scoring='roc_auc', cv = 5)
    ...: print('AUC {}, Average AUC {}'.format(scores, scores.mean()))
    ...: for n_trees in range(1, 200, 10):
    ...:     model = RandomForestClassifier(n_estimators = n_trees)
    ...:     scores = cross_val_score(model, X, y, scoring='roc_auc', cv= 5)
    ...:     print('n trees: {}, CV AUC {}, Average AUC {}'.format(n_trees, scores,
scores.mean()))
AUC [0.51916221 0.56069231 0.60930941 0.52449776 0.56386225], Average AUC
0.5555047889013732
n trees: 1, CV AUC [0.51101144 0.55995917 0.56330861 0.51144674 0.55722174],
Average AUC 0.540589540722368
n trees: 11, CV AUC [0.5100483  0.56891496 0.60347306 0.53160861 0.54732636],
Average AUC 0.5522742580427591
n trees: 21, CV AUC [0.51635904 0.54956587 0.61045943 0.51521896 0.55636443],
Average AUC 0.5495935455897505
n trees: 31, CV AUC [0.52202289 0.56930309 0.61540452 0.51558029 0.5666812 ],
Average AUC 0.5577983952557007
```

4. Results and Interpretation

- We found that depending on the maximum number of playable sets the models changes their predictive power. The results of longer matches (5 sets) are in general better predicted using any of the three models (logistic regression, decision trees or random forest).
- We can see from above that the best performing model is Logistic regression in both cases, where for Best_of = 5, it has an AUC of 0.7, for Best_of = 3 it has an AUC of 0.64
- The accuracy is not ideal but for 5-set matches it is "acceptable" and could be improved by considering other factors such as different of age between players [2].
- The (log-) ranking difference is by far the most important feature. As hoped, the difference of the log is the important function and has been verified in both cases.
- The likelihood of wins is weakly dependent on type of surface. The dependence on rounds is slightly stronger but with no obvious general behavior. For 5 sets, upsets are uncommon on the semifinals; for 3 sets the 4th round seems to stand out and there is a tendency of uniformization of the win frequency as the tournament ends. Often, if it is beyond quarterfinals, then the higher ranked player is going to win.
-

5. Summary and conclusions

This project was a step in the sports forecasting domain, understanding the features and predicting the winner, one at a time. The results give us a brief about the importance of the log difference function, and which ML classifier works the best for less data, in this situation. The current biggest application lies with the betting domain and the online games of managing teams. If explored further, this can turn out to be a financially lucrative prospect.

Future Scope:

- Include variables about the players' past performance in the short-term (last year, say). Additional factors to consider, such as motivation and home bias.
- Increase the data using data augmentation techniques, or by joining different tennis datasets. Like Women's ATP tennis stats.
- Test with other ML algorithms like ANN and SVM.

- As mentioned above the inclusion of age difference might improve the accuracy.
- Include comparisons between each two players based on recent previous confrontation.

6. References

[1] Michael Sipko, "Machine Learning for the Prediction of Professional Tennis Matches", Imperial College London MEng Computing – Final year project, June 15, 2015.

[2] Juliodel Corral JuanPrieto-Rodríguez,"Are differences in ranks good predictors for Grand Slam tennis matches?", International Journal of Forecasting, Volume 26, Issue 3, July–September 2010.

[3] https://www.betfair.com.au/hub/an-introduction-to-tennis-modelling/

[4] Eiji Konaka, "Match results prediction ability of official ATP singles ranking", arXiv:1705.05831, May 17th 2017.

[5] T. Barnett and S. R. Clarke. Combining player statistics to predict outcomes of tennis matches. IMA Journal of Management Mathematics, 16:113–120, 2005.

[6] S. Ma, C. Liu, and Y. Tan. Winning matches in Grand Slam men's singles: an analysis of player performance-related variables from 1991 to 2008. Journal of sports sciences, 2013

[7]http://www.sportspromedia.com/guest_blog/peter_webb_why_tennis_is_big_business_for_bookmakers

[8] Yang Chen, Yubo Tian, Yi Zhong, Stanford CS229 Fall 2017 - Team Project - Real Time Tennis Match Prediction Using Machine Learning