Forecasting the winner of tennis matches in the Men's ATP World Tour

Marco Tavora
DAT-NYC-41

Outline

- ✓ The goal is to estimate the likelihood of **”wins”** in tennis matches of the men’s ATP tour. A **win** happens by definition when a higher-ranked player wins the match against a lower-ranked player
- ✓ Modelling of outcomes of tennis matches has been extensively studied using a plethora of different approaches. However there are still very few works using machine learning techniques (Sipko 2015)
- ✓ This is a classification problem where the outcome is either **1** if the match result is a win or **0** if it is not
- ✓ The unit of the analysis is the match (Corral, Rodriguez 2010)

Procedure

- ✓ Define players P_1 and P_2 where the former has higher ATP ranking than the latter
- ✓ We will use as predictors:
  - ✓ Type of surface (hard, grass and clay)
  - ✓ Round of the match (from first round to the final)
  - ✓ Maximum number of playable sets (Best of 3 or Best of 5)
  - ✓ Name of the ATP Series (Grand Slam, Masters 1000, ATP 250 and ATP 500)
  - ✓ Difference between the **logarithms** of the rankings defined by **D** to take into account the non-linearity of the players quality (e.g. for bottom ranked players a difference of one position corresponds to effectively no quality difference)
- ✓ Define the binary variable **win**

$$\text{win} = \begin{cases} 1 & \text{higher ranked player wins} \\ 0 & \text{higher ranked player loses} \end{cases}$$

Procedure (ctd.)

- ✓ Subdivide the data into two groups: Best of 3 and Best of 5
- ✓ Apply models
  - ✓ Logistic regression
  - ✓ Decision tree
  - ✓ Random forest
- ✓ Measure their accuracy (using e.g. cross-validation)

Dataset and feature engineering

- ✓ Results for the men's ATP tour date back to January 2000 from the dateset http://www.tennis-data.co.uk/data.php (obtained from Kaggle)
- ✓ Our chosen features are shown below:

Date	Series	Surface	Round	Best_of	WRank	LRank
2015-12-01	ATP250	Hard	1st Round	3	44	59
2015-12-01	ATP250	Hard	1st Round	3	58	39
2015-12-01	ATP250	Hard	1st Round	3	37	51
2015-12-01	ATP250	Hard	1st Round	3	100	32
2015-12-01	ATP250	Hard	1st Round	3	113	577

- ✓ WRank: ATP Entry ranking of the match winner as of the start of the tournament
- ✓ LRank`: ATP Entry ranking of the match loser as of the start of the tournament

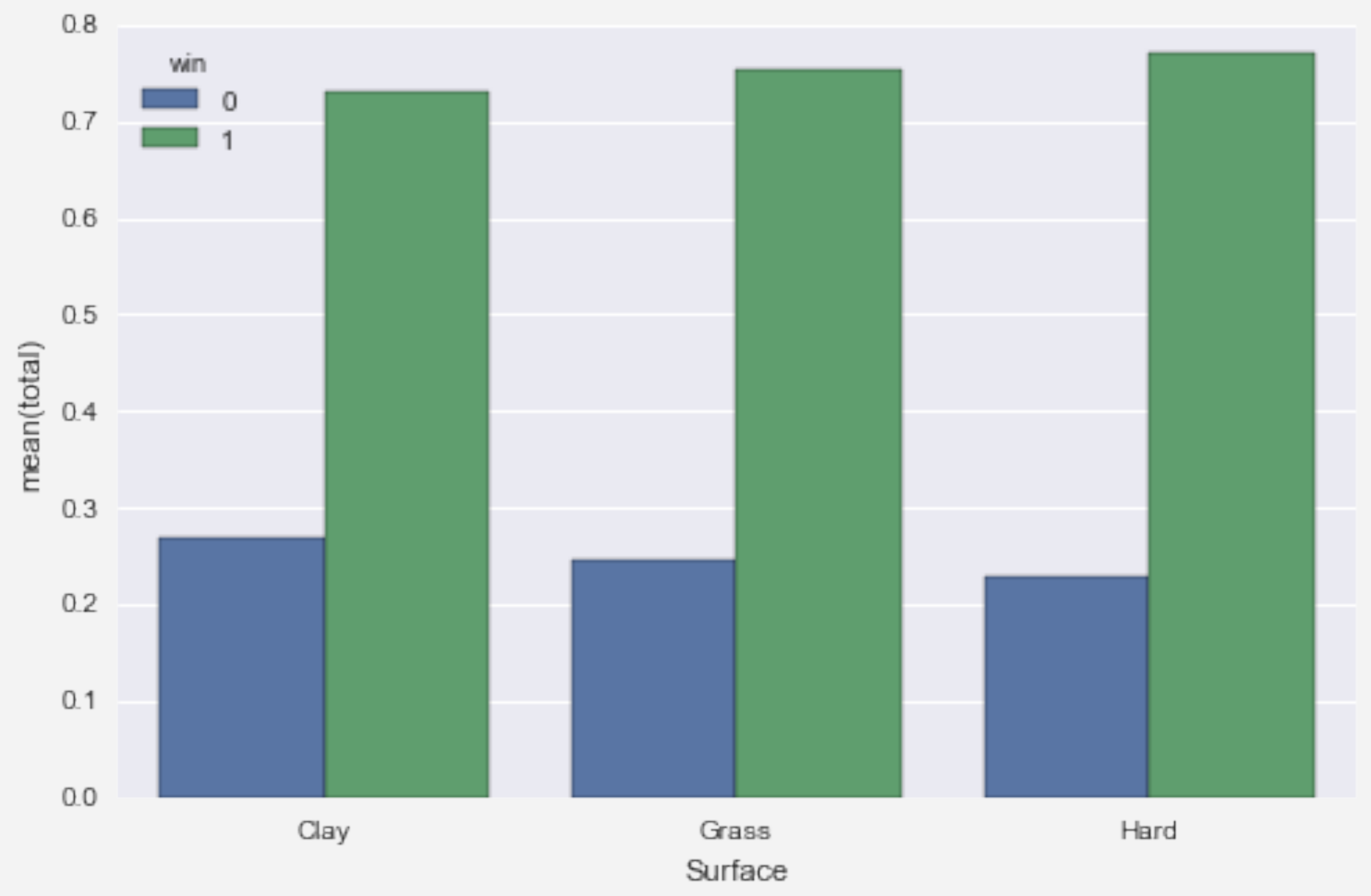
Dataset and feature engineering

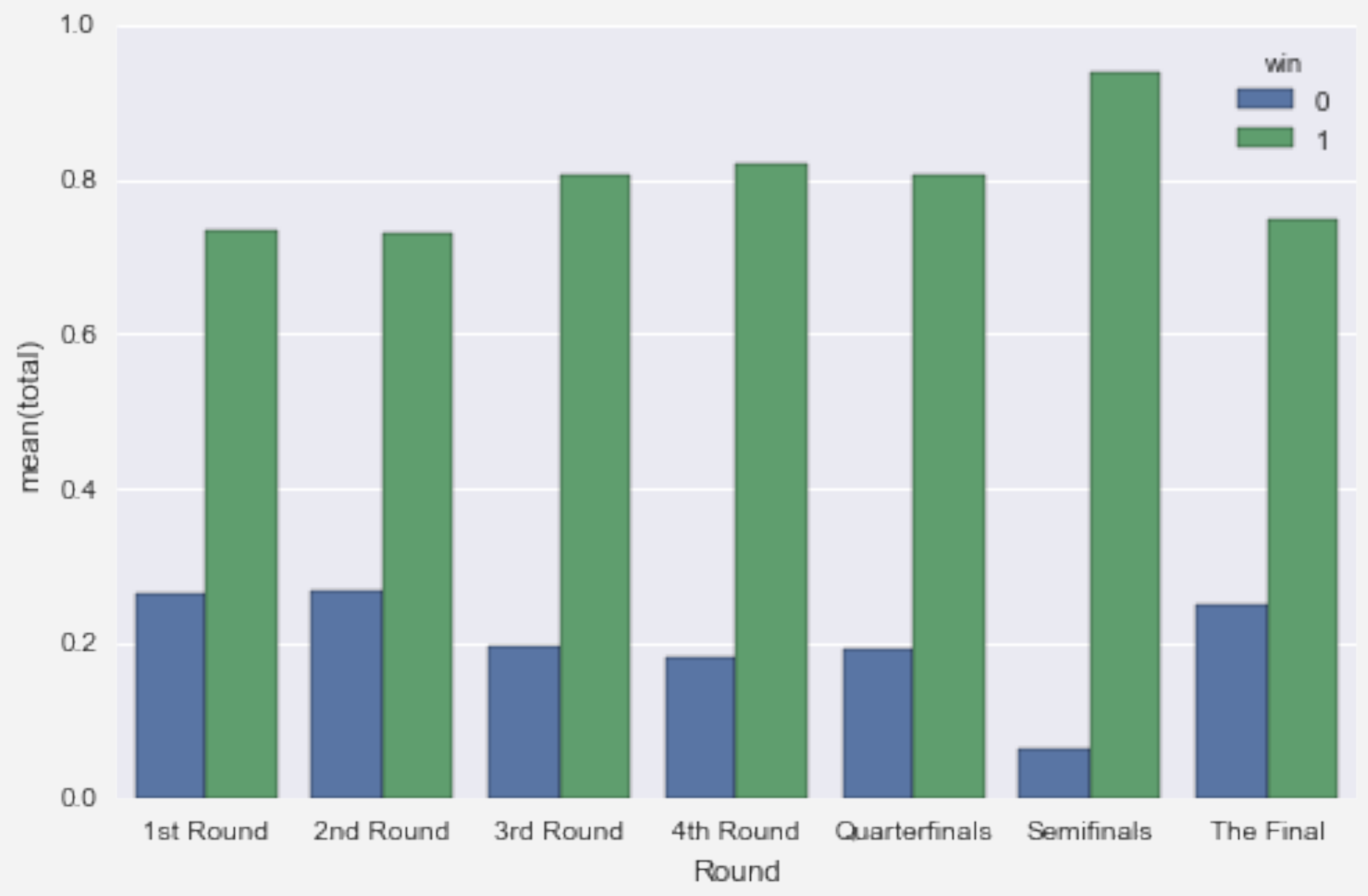
- ✓ The following modifications were performed on the data:
  - ✓ We kept only completed matches i.e. eliminated matches with injury withdrawals and walkovers
  - ✓ We used dummies for all variables except the ranking (log-)difference
  - ✓ Restricted dates to final two years only (to avoid comparing different tournaments)
  - ✓ Consider only top players (as shown by Corral, Rodriguez 2010 and verified here, this improves predictive power)
  - ✓ Stratify sample (our dataset is uneven in terms of frequency of wins and dataset must be balanced)

Exploratory Analysis – Best of 5

The frequency of upsets depends weakly on the surface type (data before stratification)

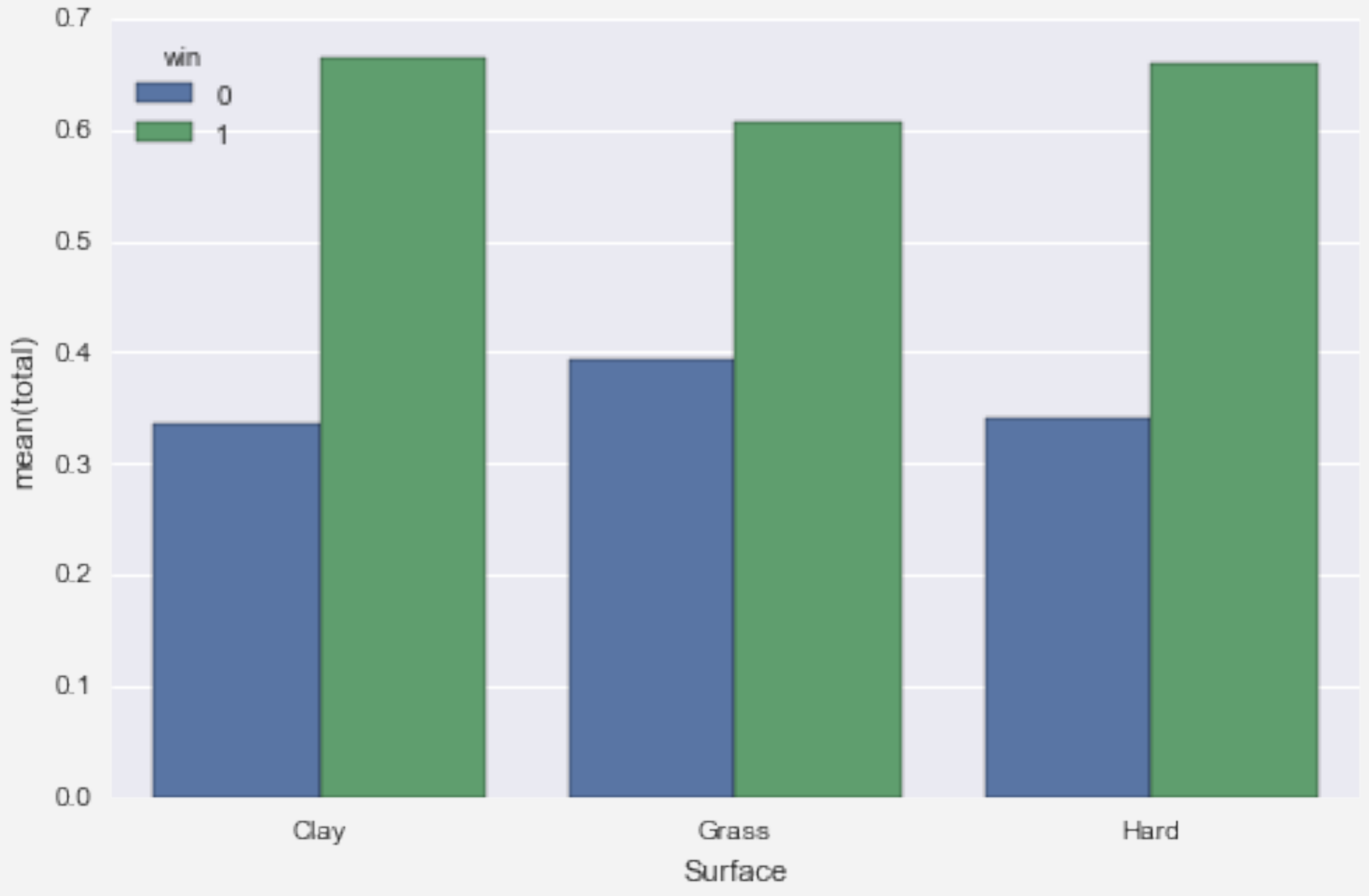
Upsets are uncommon on semifinals (data before stratification)



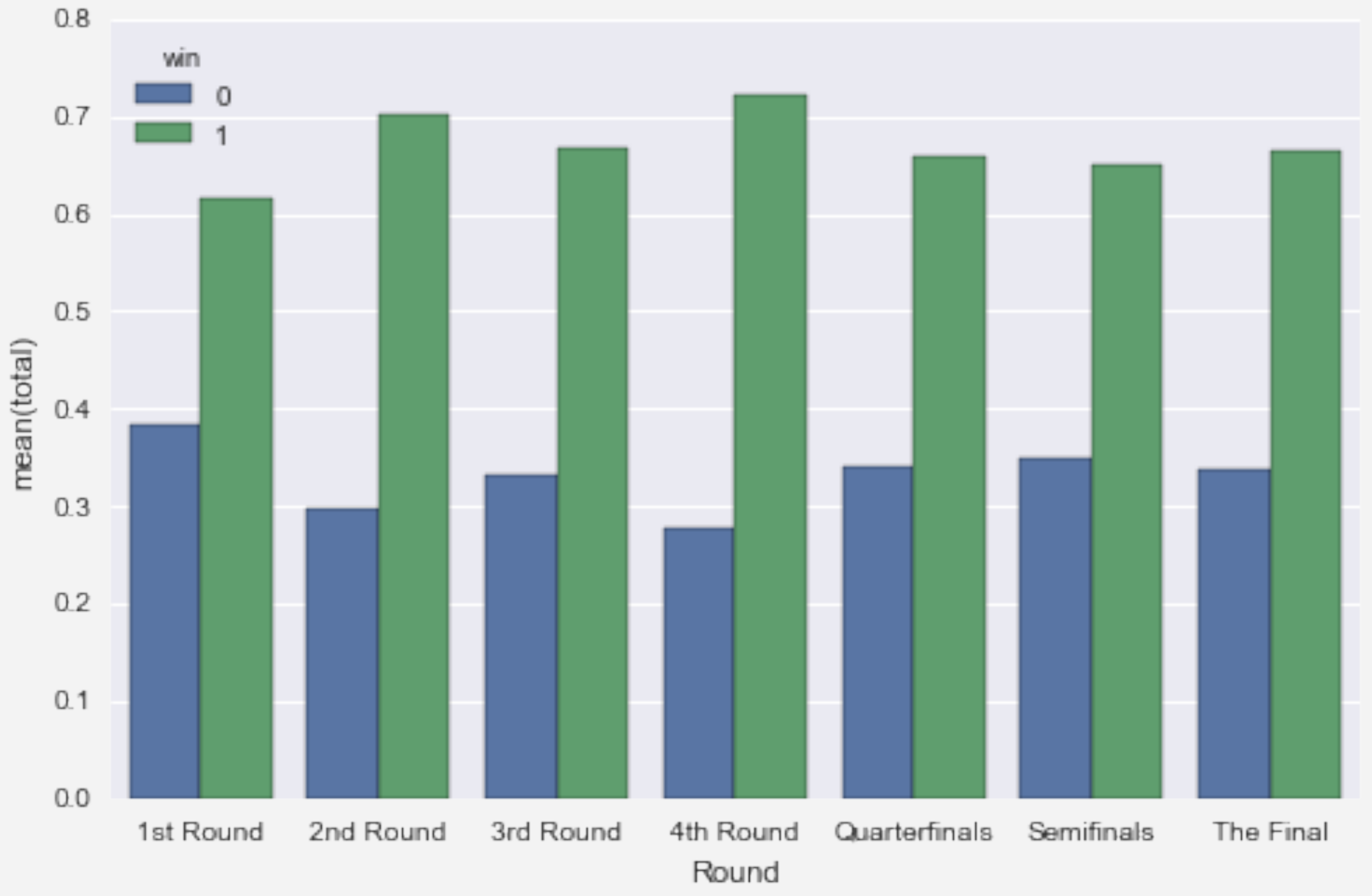


Exploratory Analysis – Best of 3

As in the case of 5 sets matches, the frequency of upsets depends weakly on the surface type (data before stratification)

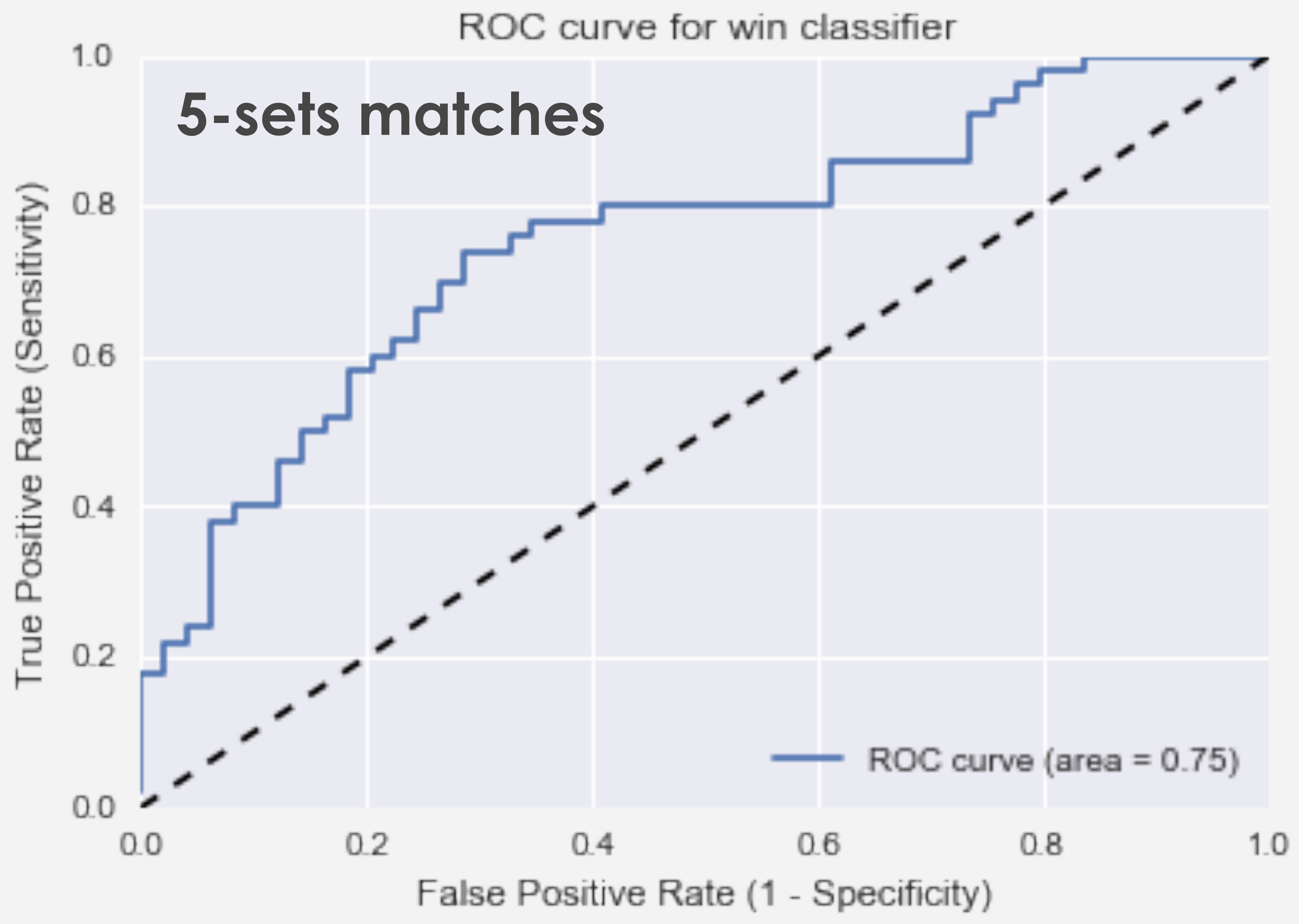


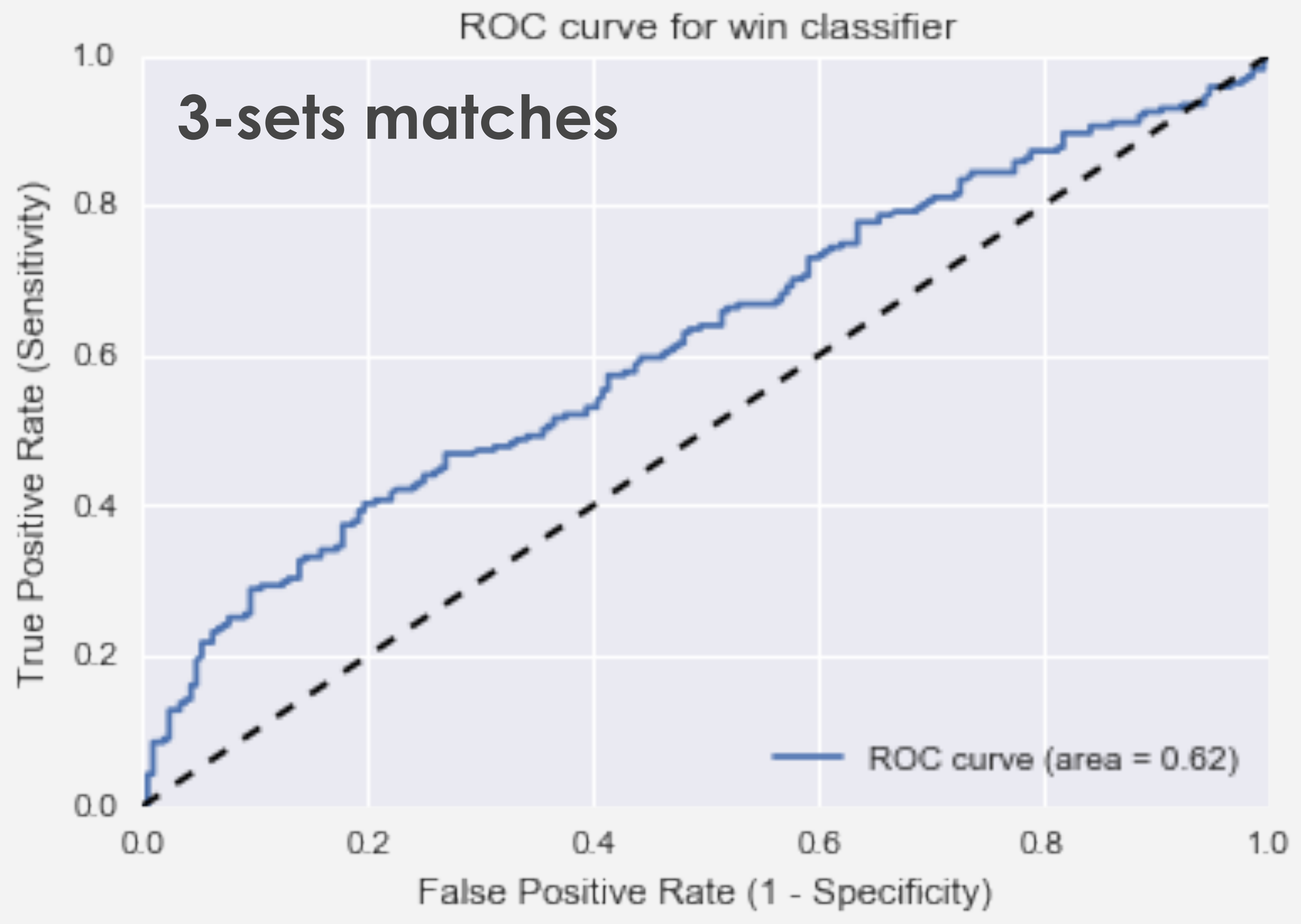
The frequency of wins is not strongly dependent on the round and tends to become more uniform at the end of the tournament



Model 1: Logistic Regression

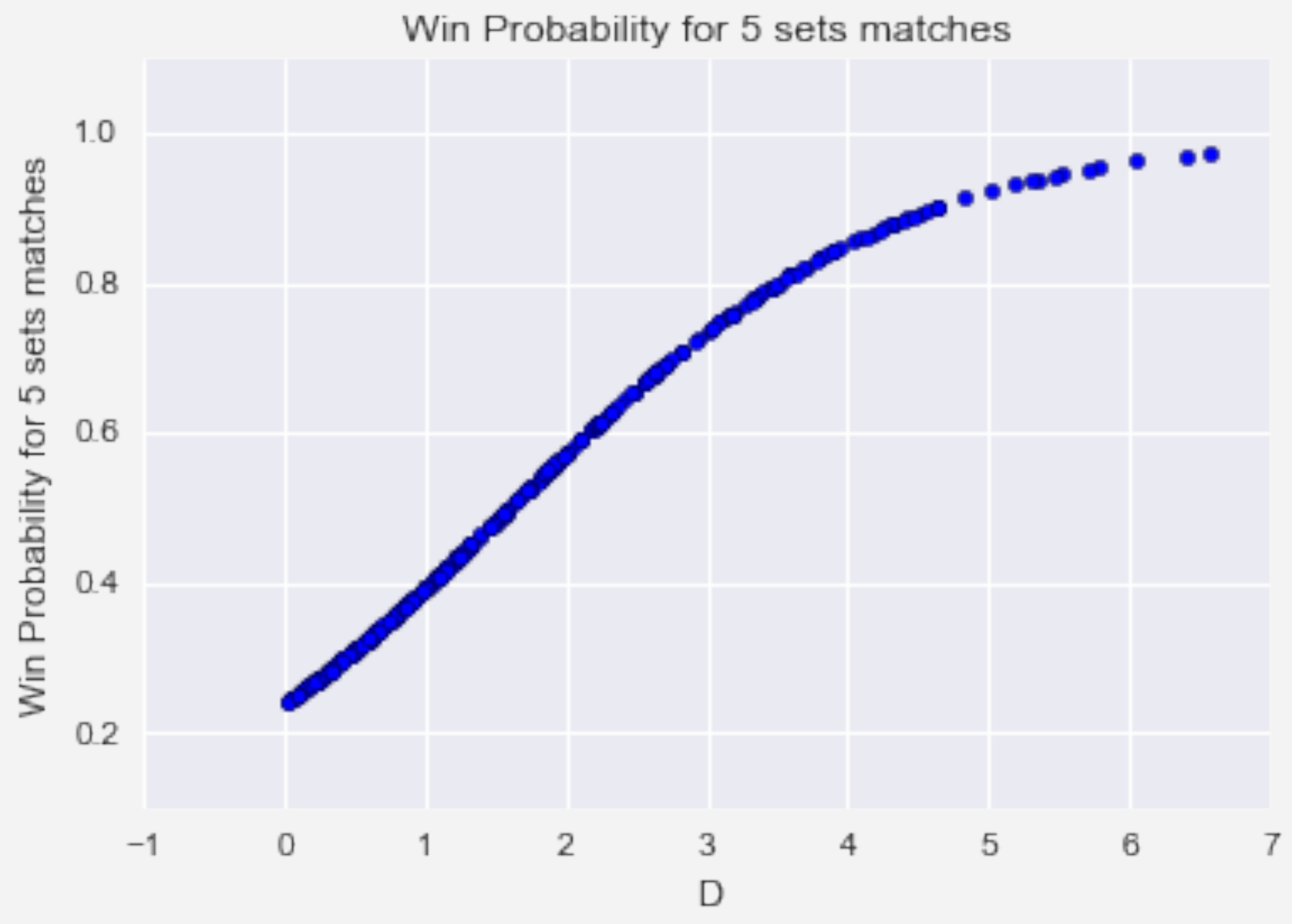
- ✓ The ROC curve shows how the sensitivity (when the actual value if true how often the prediction is correct) and specificity (when the actual value is negative, how often is the prediction correct) are affected by thresholds (a threshold is used by to convert predicted probabilities into class predictions)
- ✓ 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. We obtained AUC = 0.62 for 3-sets matches and AUC = 0.75 for 5-sets matches

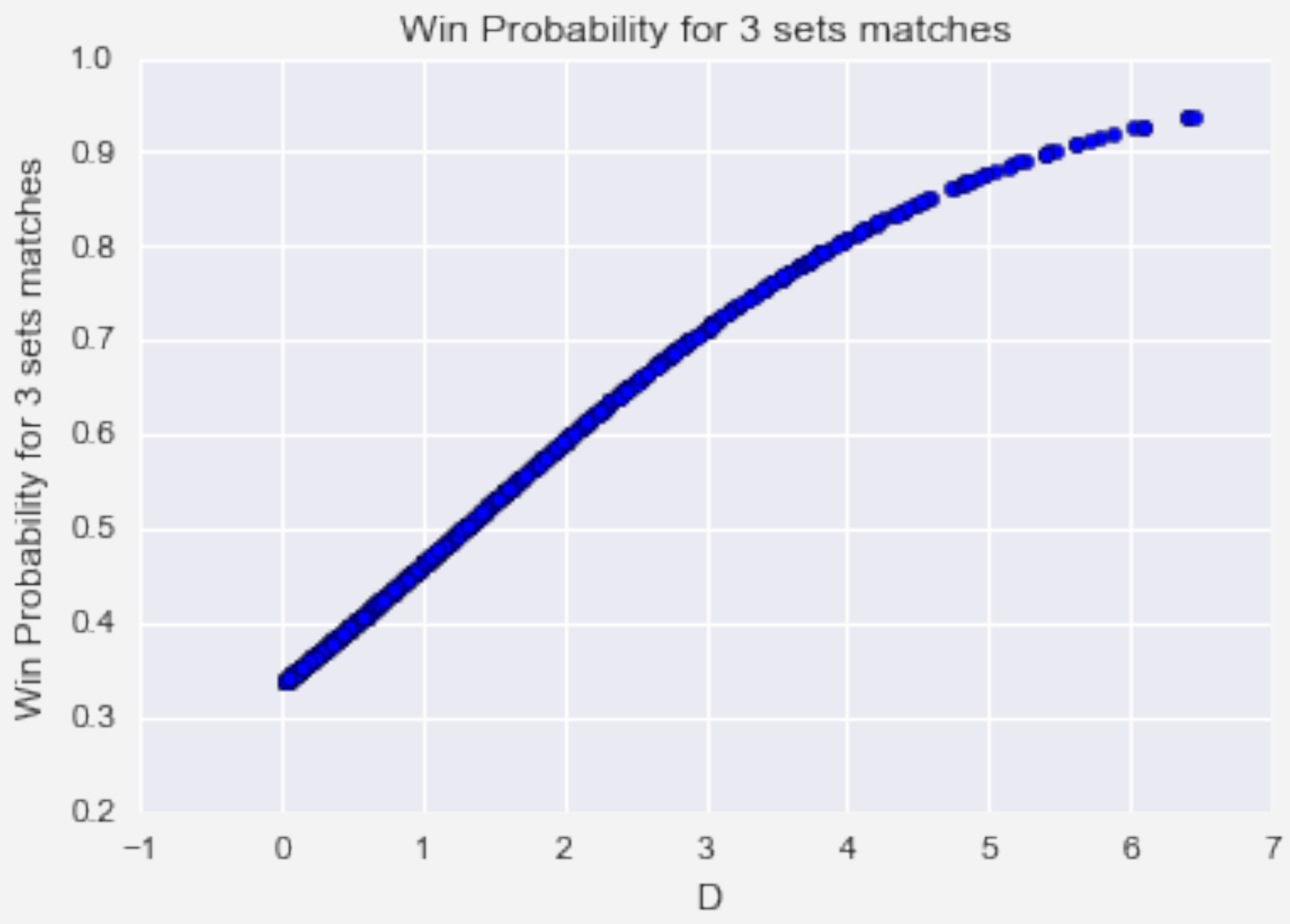




Model 1: Logistic Regression (ctd.)

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

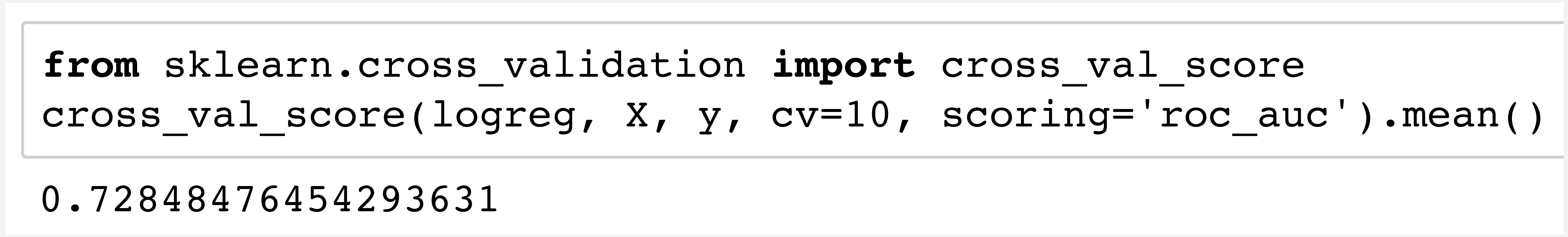




Model 1: Logistic Regression (ctd.)

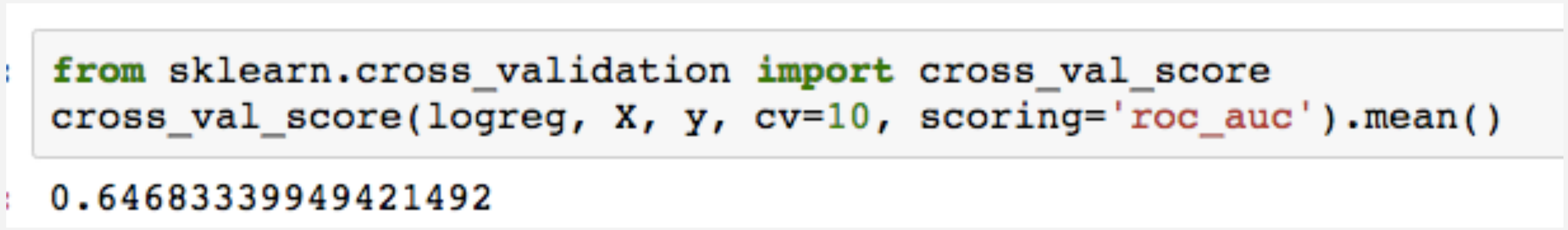
✓ We use a 10-fold cross-validation to estimate the accuracy. Cross validation is in general a reasonably accurate estimate of out-of-sample accuracy since every observation is used for both training and testing

✓ For 5 sets matches we obtain:



```
from sklearn.cross_validation import cross_val_score
cross_val_score(logreg, X, y, cv=10, scoring='roc_auc').mean()

0.72848476454293631
```

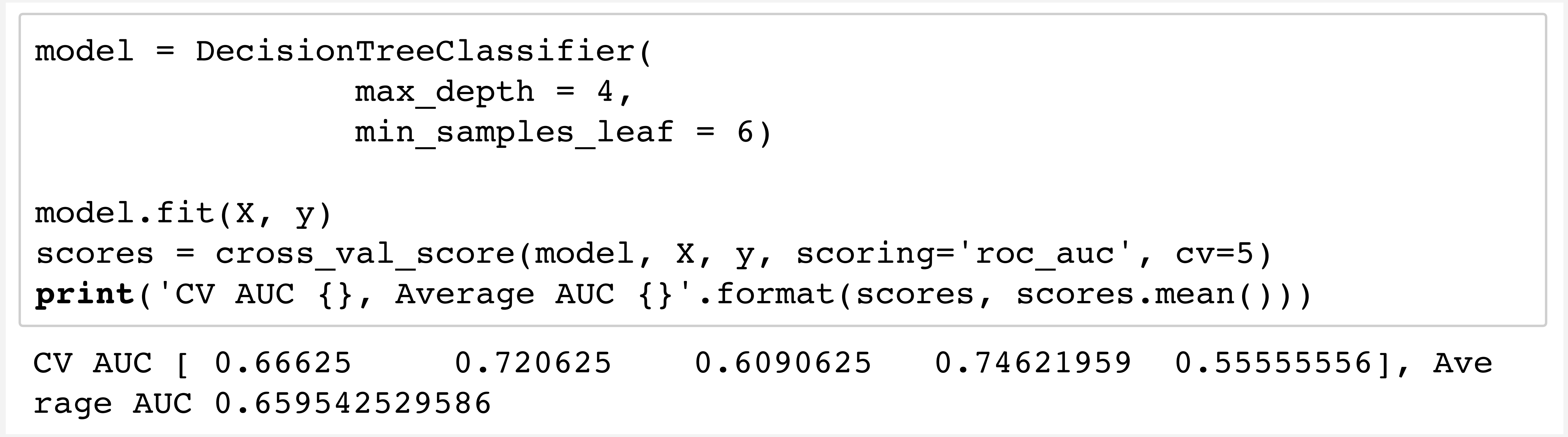
✓ For 3 sets matches we obtain:



```
from sklearn.cross_validation import cross_val_score
cross_val_score(logreg, X, y, cv=10, scoring='roc_auc').mean()

0.64683339949421492
```

Model 2: Decision Tree

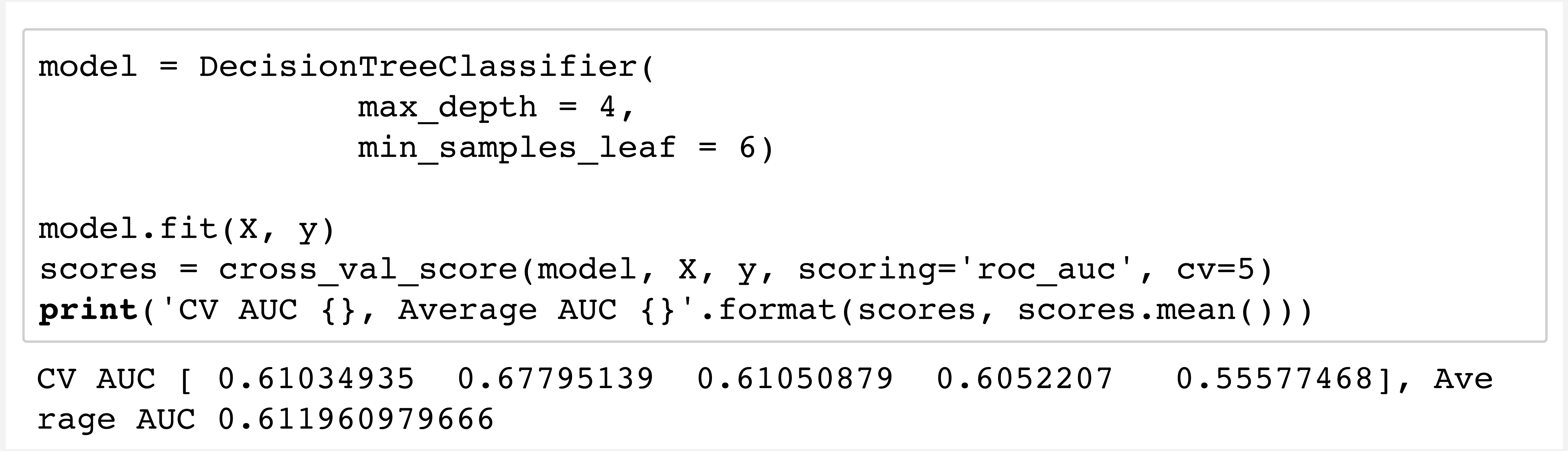
- ✓ We evaluate the decision tree using cross-validation, using AUC as the evaluation metric. We control for overfitting adjusting the maximum number of questions or minimum number of records in each final node. We use the Gini index for find the variance of classes.
- ✓ For 5 sets matches we obtain:



```
model = DecisionTreeClassifier(
                max_depth = 4,
                min_samples_leaf = 6)

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 [ 0.66625     0.720625    0.6090625   0.74621959  0.55555556], Ave
rage AUC 0.659542529586
```

- ✓ For 3 sets matches we obtain:

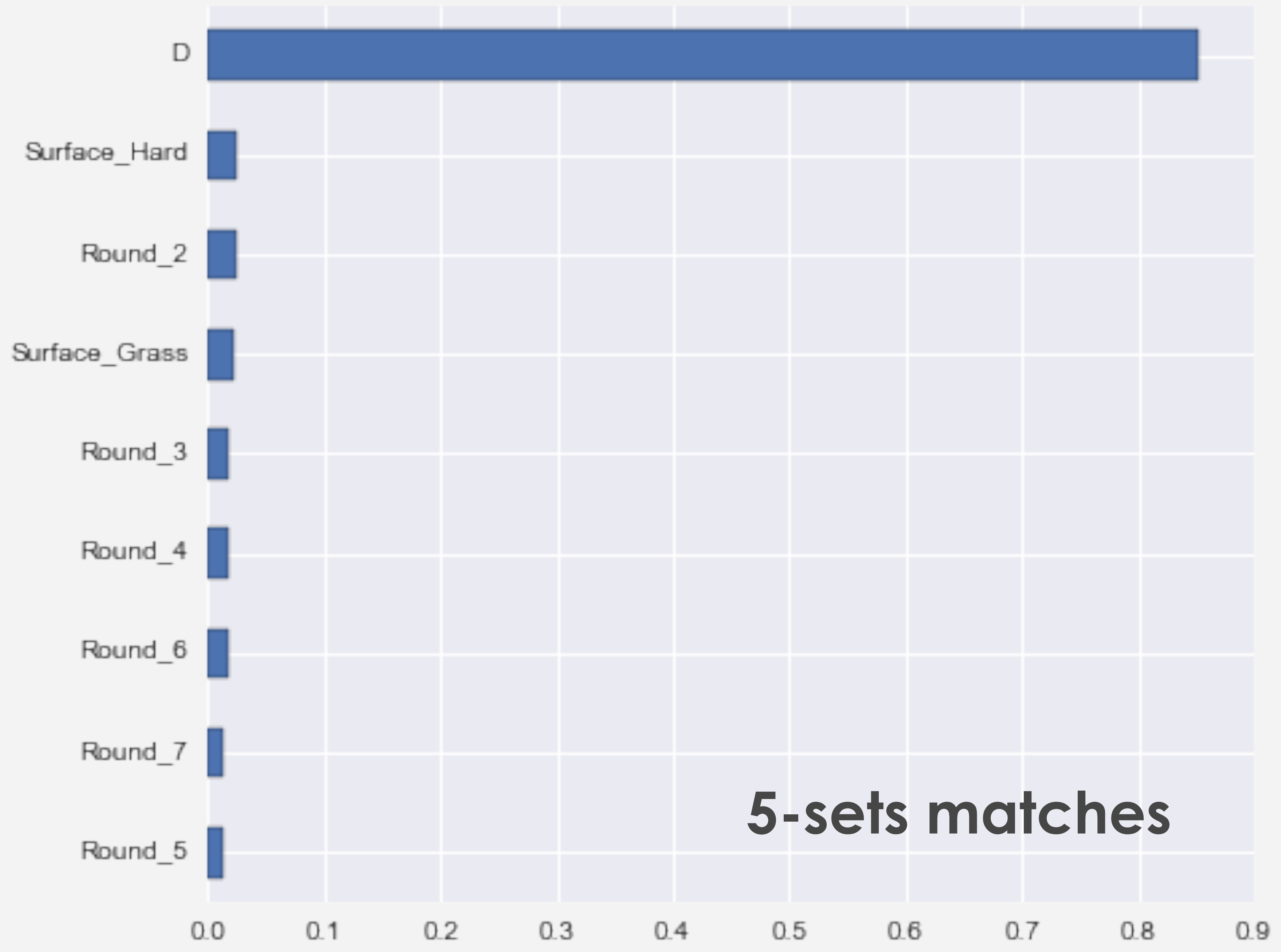


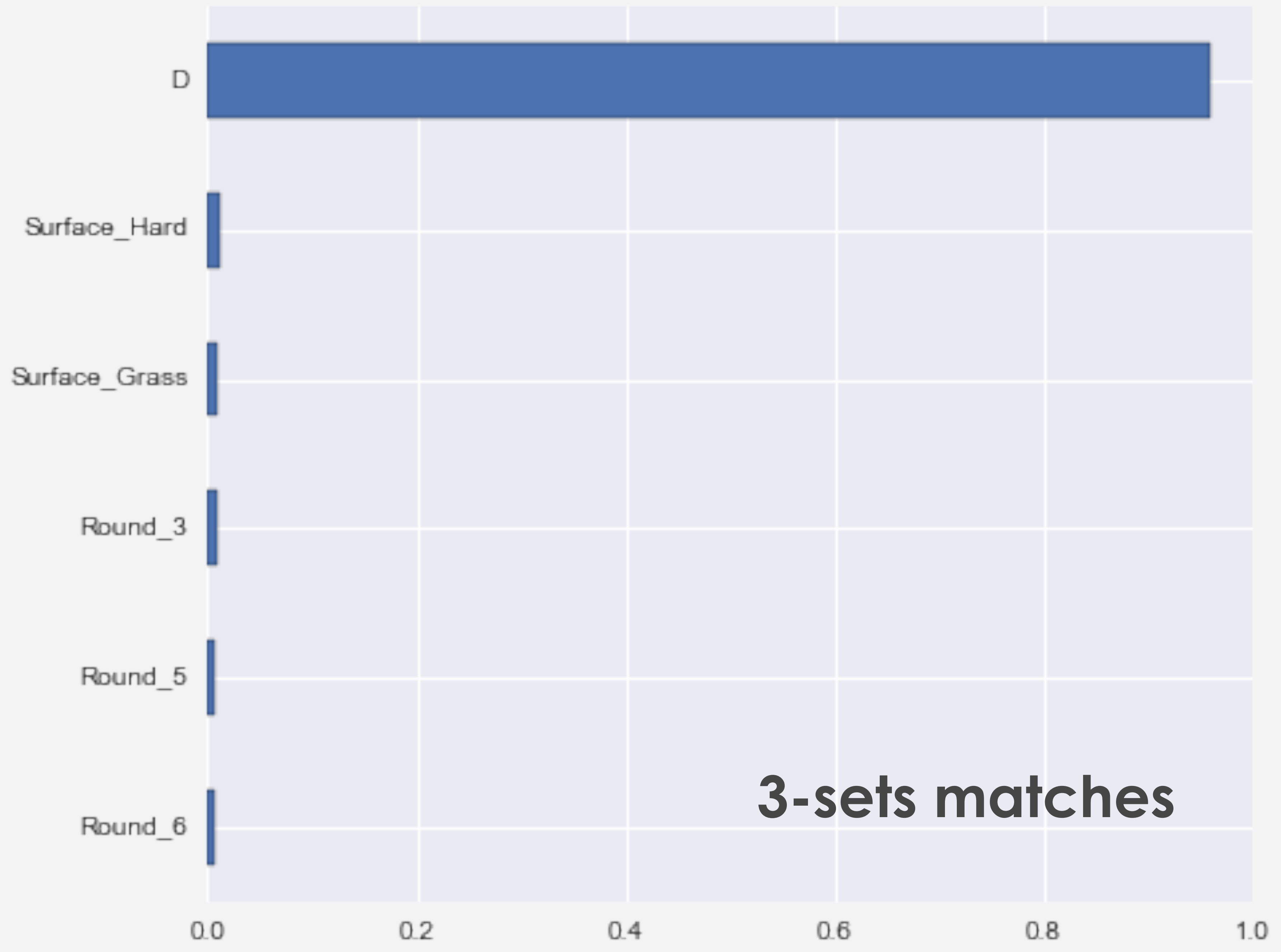
```
model = DecisionTreeClassifier(
                max_depth = 4,
                min_samples_leaf = 6)

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 [ 0.61034935  0.67795139  0.61050879  0.6052207   0.55577468], Ave
rage AUC 0.611960979666
```

Model 3: Random Forests

- ✓ Random Forests are ensembles of decision trees and have built-in protection against overfitting
- ✓ We can extract the importance of features
- ✓ We see that the ranking different is by far the most important feature!

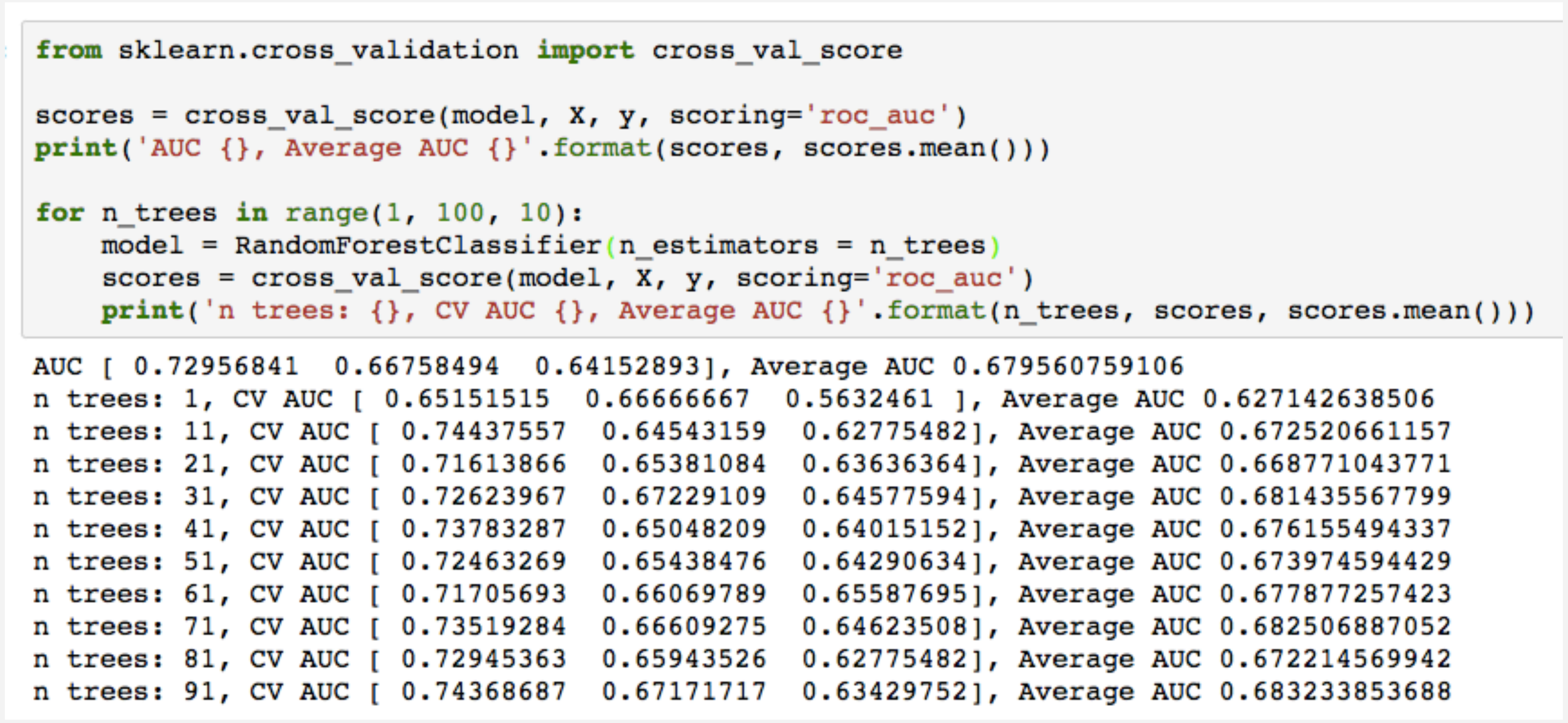




Model 3: Random Forest

- ✓ Evaluate the model using cross-validation. We increase the number of estimators to see how that improves the predictive performance.

- ✓ For 5 sets matches we obtain:

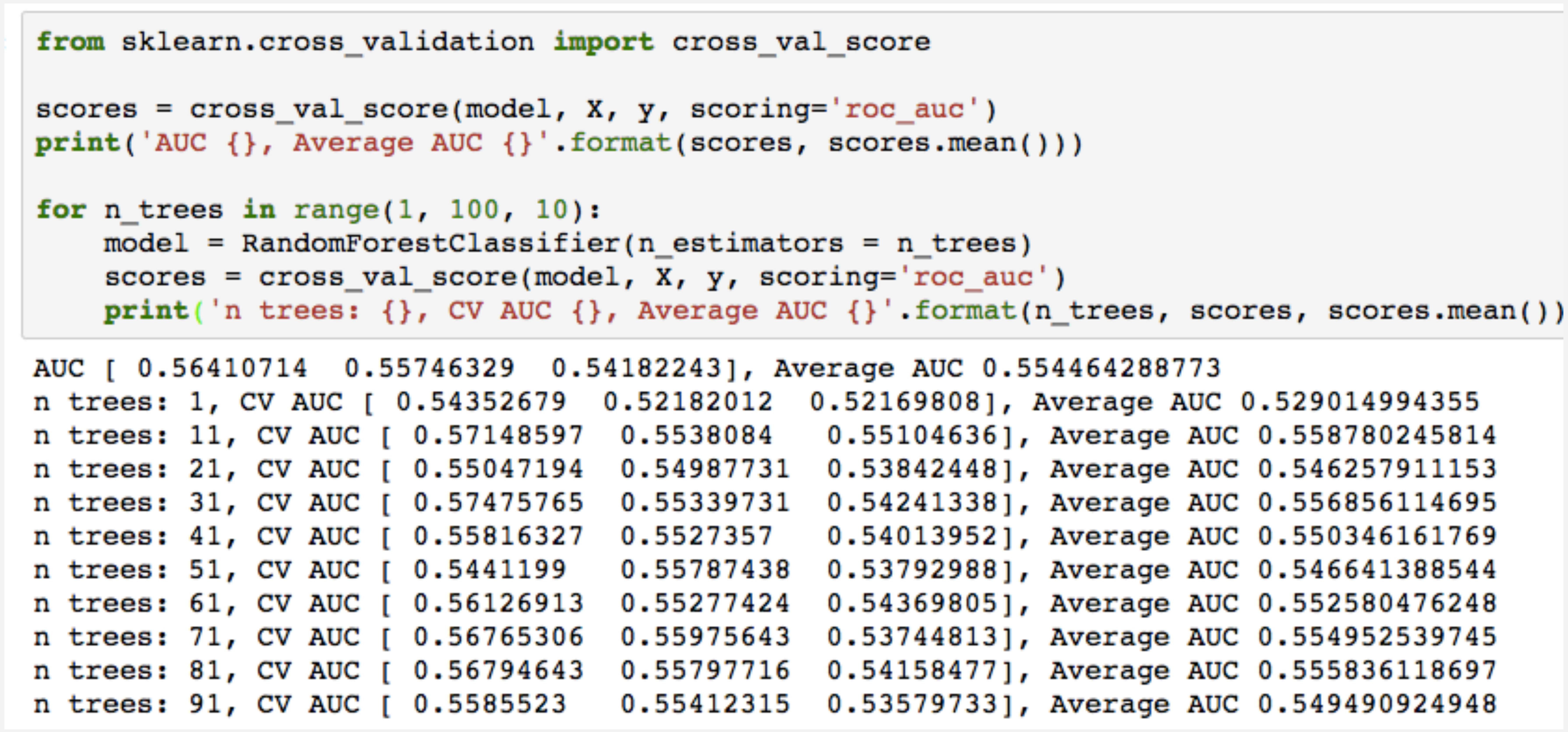


```python
from sklearn.cross_validation import cross_val_score

scores = cross_val_score(model, X, y, scoring='roc_auc')
print('AUC {}, Average AUC {}'.format(scores, scores.mean()))

for n_trees in range(1, 100, 10):
    model = RandomForestClassifier(n_estimators = n_trees)
    scores = cross_val_score(model, X, y, scoring='roc_auc')
    print('n trees: {}, CV AUC {}, Average AUC {}'.format(n_trees, scores, scores.mean()))
```

```
AUC [ 0.72956841  0.66758494  0.64152893], Average AUC 0.679560759106
n trees: 1, CV AUC [ 0.65151515  0.66666667  0.5632461 ], Average AUC 0.627142638506
n trees: 11, CV AUC [ 0.74437557  0.64543159  0.62775482], Average AUC 0.672520661157
n trees: 21, CV AUC [ 0.71613866  0.65381084  0.63636364], Average AUC 0.668771043771
n trees: 31, CV AUC [ 0.72623967  0.67229109  0.64577594], Average AUC 0.681435567799
n trees: 41, CV AUC [ 0.73783287  0.65048209  0.64015152], Average AUC 0.676155494337
n trees: 51, CV AUC [ 0.72463269  0.65438476  0.64290634], Average AUC 0.673974594429
n trees: 61, CV AUC [ 0.71705693  0.66069789  0.65587695], Average AUC 0.677877257423
n trees: 71, CV AUC [ 0.73519284  0.66609275  0.64623508], Average AUC 0.682506887052
n trees: 81, CV AUC [ 0.72945363  0.65943526  0.62775482], Average AUC 0.672214569942
n trees: 91, CV AUC [ 0.74368687  0.67171717  0.63429752], Average AUC 0.683233853688
```

Model 3: Random Forest

✓ For 3 sets matches we obtain:



```
from sklearn.cross_validation import cross_val_score

scores = cross_val_score(model, X, y, scoring='roc_auc')
print('AUC {}, Average AUC {}'.format(scores, scores.mean()))

for n_trees in range(1, 100, 10):
    model = RandomForestClassifier(n_estimators = n_trees)
    scores = cross_val_score(model, X, y, scoring='roc_auc')
    print('n trees: {}, CV AUC {}, Average AUC {}'.format(n_trees, scores, scores.mean())
```

```
AUC [ 0.56410714  0.55746329  0.54182243], Average AUC 0.554464288773
n trees: 1, CV AUC [ 0.54352679  0.52182012  0.52169808], Average AUC 0.529014994355
n trees: 11, CV AUC [ 0.57148597  0.5538084   0.55104636], Average AUC 0.558780245814
n trees: 21, CV AUC [ 0.55047194  0.54987731  0.53842448], Average AUC 0.546257911153
n trees: 31, CV AUC [ 0.57475765  0.55339731  0.54241338], Average AUC 0.556856114695
n trees: 41, CV AUC [ 0.55816327  0.5527357   0.54013952], Average AUC 0.550346161769
n trees: 51, CV AUC [ 0.5441199   0.55787438  0.53792988], Average AUC 0.546641388544
n trees: 61, CV AUC [ 0.56126913  0.55277424  0.54369805], Average AUC 0.552580476248
n trees: 71, CV AUC [ 0.56765306  0.55975643  0.53744813], Average AUC 0.554952539745
n trees: 81, CV AUC [ 0.56794643  0.55797716  0.54158477], Average AUC 0.555836118697
n trees: 91, CV AUC [ 0.5585523   0.55412315  0.53579733], Average AUC 0.549490924948
```

Conclusions and extensions

- ✓ 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 regressios, decision trees or random forest).

- ✓ 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 (Corral, Rodriguez 2010)

- ✓ The (log-) ranking difference is by far the most important feature

- ✓ The likelihood of wins is weakly dependent on type of surface. The dependence on rounds is slighly 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

- ✓ Next steps would be to:

  - ✓ Include variables about the players' past performance in the short-term (last year, say)
  - ✓ As mentioned above the inclusion of age difference might improve the accuracy
  - ✓ Include comparisons between each two players based on recent previous confrontation
  - ✓ Take into account time effects by some type of discount factor