MACHINE LEARNING PROJECT

ON

Tennis Match Prediction

Depends on Past Data & On-Match Data

Group Members:

JeetSaha, Academy Of Technology, 151690110145

Sanmitra Kumar, Academy Of Technology, 151690110165

Sushmita Shaw, Academy Of Technology, 151690110187

Arijit Chowdhury, Academy Of Technology, 151690110018

Table of Contents

Abstract

Our project had two main objectives. First, we wanted to use historical tennis match data to predict the outcomes of future tennis matches. Next, we wanted to use the predictions from our resulting model to beat the current betting odds.

Acknowledgement

We would like to take this opportunity to express our profound gratitude and deep regards to our faculty **Sir. Titas Roychowdhury** for his exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessing, help and guidance given by him time to time shall carry us a long way in the journey of life on which we are about to embark.

Each of us project members have worked to our utmost best as to make this project a success in our respective fields. We are grateful to one another for the cooperation during the course of this assignment.

Jeet Saha

Sanmitra Kumar

Sushmita Shaw

Arijit Chowdhury

Project Objective:-

Tennis is an international sport, enjoyed by fans in countries all over the world.
Unsurprisingly, professional tennis players come from an equally diverse background, drawn from countries throughout North America, South America, Europe and Asia. From each of these regions players come equipped with different playing styles and specialties. In addition to this, tennis fans know that the sport is played on three unique surfaces (clay, grass, and hard courts), each lending itself to different play strategies. There are a huge number of variables that define each and every tennis match, making the sport both exciting and unpredictable. Being tennis fans ourselves, we decided to move away from our gut instincts and take a new approach to predicting the outcomes of our favorite matches.

Main objective is to predict the outcome of the match result before the the final result with good accuracy with the uses of different models and serve the result.

Project Scope

There are two main categories of tennis betting: pre-game and in-game, with the distinction that pregame bets cannot be placed after the game commences. Furthermore, it is usually possible to bet on a variety of factors, such as the winner of the match, the score of different sets, the total number of games, etc. We will focus on pre-game bets on the winner of the match, as the odds for this bet type are most available historically, allowing us to perform a more comprehensive evaluation of the performance of our model against the betting market. Bets on tennis matches can be placed either with bookmakers or on betting exchanges. Traditional bookmakers (e.g., Pinnacle Sports) set odds for the different outcomes of a match, and a bettor competes against the bookmakers. In the case of betting exchanges (e.g., Bet fair), customers can bet against odds set by other customers. The exchange matches the customers' bets to earn a risk-free profit by charging a commission on each bet matched.

Requirement Specifications

- Using the results for the men’s ATP tour data back to January 2000,including Grand Stames ,Mater Series ,Mater Cup and International Series competitions.
- Historical data is used to predict who will be the match winner.
- **tourney_id:**
  This attribute holds the tournament id.
- **tourney_name:**
  This attributes has the records of tournaments name
- **surface:**
  It holds the surface record, there we have four different surfaces(clay,hard,grass,carpet).
- **draw_size:**
- **tourney_level:**
- **tourney_date:**
  holds the date of each tournament
- **match_num:**
  Holds the match no. of every match
- **winner_id:**
  The attribute winner_id holds the id of winner of each match.
- **winner_seed:**
- **winner_entry:**
- **winner_name:**
  Winner of each match is recorded in this particular section.
- **winner_hand:**
  Whether the player is right handed or left handed, the particular information is stored in this particular section.
- **winner_ht:**
  Winner’s height is stored in this section.
- **winner_ioc:**
- **winner_age:**
  Winner’s age is recorded here.

- **winner_rank:**
  Different player has its own rank. So the rank is stored in this section.
- **winner_rank_points:**
  Points for rank is stored in this section.
- **loser_id:**
  The attribute loser_id holds the id of winner of each match.
- **loser_seed:**
- **loser_entry:**
- **loser_name:**

Loser of each match is recorded in this particular section.

- **loser_hand:**

Whether the loser is right handed or left handed, the particular information is stored in this particular section.

- **loser_ht:**

loser’s height is stored in this section.

- **loser_ioc:**
- **loser_age:**

Loser’s age is recorded here.

- **loser_rank:**

Different player has its own rank. So the loser’s rank is stored in this section.

- **loser_rank_points:**

Points for rank of loser is stored in this section.

- **score:**

Score of each match

- **Best_of:**
- **Round:**
- **Minutes:**

Total time taken for a match

- **W_ace:**

In match result

- **W_df:**

In match result

- **w_svpt:**

In match result

- **w_1stIn:**

in match result

- **w_1stWon:**

In match result

- **w_2ndWon:**

In match result

- **w_SvGms:**

In match result

- **w_bpSaved:**

In match result

- **w_bpFaced:**

In match result

- **l_ace:**

In match result

- **l_df:**

In match result

- **l_svpt:**

In match result

- **l_1stIn:**

In match result

- **l_1stWon:**

In match result

- **l_2ndWon:**

In match result

- **l_SvGms:**

In match result

- **l_bpSaved:**

In match result

- **l_bpFaced:**

In match result

ARRANGEMENT OF THE DATA :

Here we get a match information where the winner and loser is declared already at the beginning. That's why we break the date set into two parts. One w.r.t the winner and another w.r.t the loser, and rename it as player & his opponent. So if a player is manage to win a match against his opponent the value of the win(newly introduced column) columns is 1 else 0.

That's why every time we do such operations we take the modified data set to get an accurate and fair result.

APPROACH TO PROBLEM SOLUTION

❖ MACHINE LEARNING ALGORITHMS

➢ <u>Logistic Regression</u>

In logistic regression, we have a hypothesis of the form:

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1+e^{-\theta T x}} \quad ,$$

where g is the logistic function.

We assume that the binary classification labels are drawn from a distribution such that:

P(y = 1 | x; θ) = h_θ(x)

P(y = 0 | x; θ) = 1 − h_θ (x)

Given a set of labelled set of training examples, we choose θ to maximize the log-likelihood:

$$l(\theta) = \sum_i y^{(i)}\, logh\left(x^{(i)}\right) + \left(1 - y^{(i)}\right) log\left(1 - h\left(x^{(i)}\right)\right)$$

We can maximize the log-likelihood by stochastic gradient ascent under which the update rule is the following (where α is the learning rate) and we run until the update is smaller than a certain threshold:

$$\theta = \theta + \alpha(y^{(i)} - h_\theta\left(x^{(i)}\right)x^{(i)}$$

- **<u>K-NEAREST NEIGHBORS CLASSIFICATION</u>**

  K nearest neighbors is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions). KNN has been used in statistical estimation and pattern recognition already in the beginning of 1970's as a non-parametric technique.

  A case is classified by a majority vote of its neighbors, with the case being assigned to the class most common amongst its K nearest neighbors measured by a distance function. If K = 1, then the case is simply assigned to the class of its nearest neighbor.

DISTANCE FORMULA

EUCLIDEAN $\sqrt{\sum_{\mathbf{i=1}}^{\mathbf{k}}(\mathbf{x_{i}}_{-}\mathbf{y_i})^2}$

It should also be noted that all three distance measures are only valid for continuous variables. In the instance of categorical variables the Hamming distance must be used. It also brings up the issue of standardization of the numerical variables between 0 and 1 when there is a mixture of numerical and categorical variables in the dataset.

$$D^H = \sum_{i=1}^{k} |x_i - y_i|$$
$$x = y => D = 0$$
$$x \neq y => D = 1$$

Choosing the optimal value for K is best done by first inspecting the data. In general, a large K value is more precise as it reduces the overall noise but there is no guarantee. Cross-validation is another way to retrospectively determine a good K value by using an independent dataset to validate the K value. Historically, the optimal K for most datasets has been between 3-10. That produces much better results than 1NN.

➢ NAIVE BAYES CLASSIFIER

Naive Bayes classifiers can handle an arbitrary number of independent variables whether continuous or categorical. Given a set of variables, X = {x1,x2,x...,xd}, we want to construct the posterior probability for the event Cj among a set of possible outcomes C = {c1,c2,c...,cd}. In a more familiar language, X is the predictors and C is the set of categorical levels present in the dependent variable. Using Bayes' rule:

$$p(C_j \mid x_1, x_2, \ldots, x_d) \propto p(x_1, x_2, \ldots, x_d \mid C_j) p(C_j)$$

where p(Cj | x1,x2,x...,xd) is the posterior probability of class membership, i.e., the probability that X belongs to Cj. Since Naive Bayes assumes that the conditional probabilities of the independent variables are statistically independent we can decompose the likelihood to a product of terms:

$$p(X \mid C_j) \propto \prod_{k=1}^{d} p(x_k \mid C_j)$$

and rewrite the posterior as:

$$p(C_j \mid X) \propto p(C_j) \prod_{k=1}^{d} p(x_k \mid C_j)$$

Using Bayes' rule above, we label a new case X with a class level Cj that achieves the highest posterior probability.

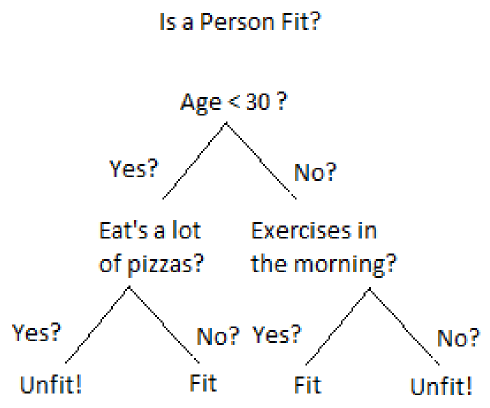
Although the assumption that the predictor (independent) variables are independent is not always accurate, it does simplify the classification task dramatically, since it allows the class conditional densities p(xk | Cj) to be calculated separately for each variable, i.e., it reduces a multidimensional task to a number of one-dimensional ones. In effect, Naive Bayes reduces a high-dimensional density estimation task to a one-dimensional kernel density estimation. Furthermore, the assumption does not seem to greatly affect the posterior probabilities, especially in regions near decision boundaries, thus, leaving the classification task unaffected.

Naive Bayes can be modeled in several different ways including normal, lognormal, gamma and Poisson density functions:

$$p(x_k \mid C_j) = \left\{ \begin{array}{lll} \dfrac{1}{\sigma_{kj}\sqrt{2\pi}} \exp\left(\dfrac{-(x - \mu_{kj})^2}{2\sigma_{kj}} \right), & -\infty < x < \infty, -\infty < \mu_{kj} <, \sigma_{kj} > 0 & \text{Normal} \\ \mu_{kj} : \text{mean}, \ \sigma_{kj} : \text{standard deviation} & & \\ \dfrac{1}{x\sigma_{kj}(2\pi)^{1/2}} \exp\left\{ \dfrac{-[\log(x/m_{kj})]^2}{2\sigma_{kj}{}^2} \right\}, & 0 < x < \infty, m_{kj} > 0, \sigma_{kj} > 0 & \text{Lognormal} \\ m_{kj} : \text{scale parameter}, \ \sigma_{kj} : \text{shape parameter} & & \\ \dfrac{\left(\dfrac{x}{b_{kj}} \right)^{c_{kj}-1}}{b_{kj}\Gamma(c_{kj})} \exp\left(\dfrac{-x}{b_{kj}} \right), & 0 \le x < \infty, b_{kj} > 0, c_{kj} > 0 & \text{Gamma} \\ b_{kj} : \text{scale parameter}, \ c_{kj} : \text{shape parameter} & & \\ \dfrac{\lambda_{kj} \exp(-\lambda_{kj})}{x!}, & 0 \le x < \infty, \lambda_{kj} > 0, x = 0, 1, 2, \ldots & \text{Poisson} \\ \lambda_{kj} : \text{mean} & & \end{array} \right\}$$

➢ Decision Tree

Decision Trees are a type of Supervised Machine Learning (that is you explain what the input is and what the corresponding output is in the training data) where the data is continuously split according to a certain parameter. The tree can be explained by two entities, namely decision nodes and leaves. The leaves are the decisions or the final outcomes. And the decision nodes are where the data is split.



- **Entropy**

  Entropy, also called as Shannon Entropy is denoted by H(S) for a finite set S, is the measure of the amount of uncertainty or randomness in data.

  $$H(S) = \sum_{x \in X} p(x) \log_2 \frac{1}{p(x)}$$

  Intuitively, it tells us about the predictability of a certain event. Example, consider a coin toss whose probability of heads is 0.5 and probability of tails is 0.5. Here the entropy is the highest possible, since there's no way of determining what the outcome might be. Alternatively, consider a coin which has heads on both the sides, the entropy of such an event can be predicted perfectly since we know beforehand that it'll always be heads. In other words, this event has **no randomness**hence it's entropy is zero.

In particular, lower values imply less uncertainty while higher values imply high uncertainty.

- **Information Gain**

Information gain is also called as Kullback-Leibler divergence denoted by IG(S,A) for a set S is the effective change in entropy after deciding on a particular attribute A. It measures the relative change in entropy with respect to the independent variables.

$$IG(S, A) = H(S) - H(S, A)$$

Alternatively,

$$IG(S, A) = H(S) - \sum_{i=0}^{n} P(x) * H(x)$$

where IG(S, A) is the information gain by applying feature A. H(S) is the Entropy of the entire set, while the second term calculates the Entropy after applying the feature A, where P(x) is the probability of event x.

Based on the above mentioned algorithms we designed our models and found out the confusion matrix as well as we calculated the accuracy, precision, recall & f-measure scores.

▪ <u>Confusion Matrix</u>

The confusion matrix [10] is a tabular representation that provides visualization of the performance of a classification algorithm. Each column of the matrix denotes the examples in a predicted class, while each row indicates the examples in an actual class. This helps to find out any type of misclassification due to the classifier. It provides more detailed analysis than classification accuracy. Classification accuracy is not a reliable metric for assessing the performance of a classifier as it may produce misleading results when the numbers of samples in different classes vary greatly. The confusion matrix entries can be defined as follows;

i. True positive (tp) is the number of 'positive' instances categorized as 'positive'.

ii. False positive (fp) is the number of 'negative' instances categorized as 'positive'.

iii. False negative (fn) is the number of 'positive' instances categorized as 'negative'.

iv. True negative (tn) is the number of 'negative' instances categorized as 'negative'.

Predicted class / Actual Class	Positive	Negative
Positive	tp	fp
Negative	fn	tn

▪ <u>Accuracy</u>

Accuracy is defined as a ratio of sum of the instances classified correctly to the total number of instances.

$$Accuracy = \frac{tp+tn}{tp+fp+fn+tn}$$

- **<u>Precision</u>**

  Precision is defined as the ratio of correctly classified data in positive class to the total number of data classified as to be in positive class.

$$Precision = \frac{tp}{tp+fp}$$

- **<u>Recall</u>**

  Recall or TP rate is defined as the ratio of tp to the total number of instances classified under positive class.

$$Recall = \frac{tp}{tp+fn}$$

- **<u>F-measure</u>**

  F-measure is defined as a combined representation of Precision and Recall and is defined as follows:

$$F - measure = 2 * \frac{Precision * Recall}{Precision+Recall}$$

DATA PROCESSING AND FEATURE SELECTION

MODELS

We have designed three models from our given tennis dataset, those are predicting the winner based on historical data, predicting the winner based on in-game results & predicting the winner without using any Learning Algorithms simply comparing their win percentage with respect to each surface.

PREPROCESSING

We figured out all those attributes which are essential for our prediction. Considering those attributes if any attributes have categorical values we converted them into encoded integer values using LabelEncoder library function of the Preprocessing library of Scikit-Learn. If any attributes have special characters then label encoding was not possible for those attributes. Instead we encoded them by counting the unique entries and map each entries with the unique counts. There were some missing values in some attributes. We calculated the median of those attributes and replaced those missing values with the median value. The dataset was divided into training set(80% records) & testing set(20% records).

- **Feature Selection**
  - **Model Based On Historical Data**
    We filtered out only the columns like winner name, winner age, winner hand, winner height, winner rank, winner rank points & all the respective columns for loser also including the surface column. We created two different records for winner & loser. In the winner record all the winner attributes were labelled as 'player' & all the loser attributes were labelled as 'opponent' & a new attribute was added as 'win' containing all entries as '1' . And for the loser record all the winner attributes were labelled as 'opponent & all the loser attributes were labelled as 'player & a new attribute was added as 'win' containing all entries as '0' .
    Then both the winner & record were merged together to form our new dataset. Here our target variable is the 'win' attribute.
  - **Model Based On In-game Result**
    Here in this model we have considered only those columns which reflects the in-game results of each match. This kind of model is helpful for betting odds. We figured out all the unique players from the dataset and counted the frequency of their data in the dataset. Based on that count values we consider only 5-10 players with highest frequencies and retrieved all the records of those players and prepared a separate dataset.
    Now considering a single player suppose, 'Roger Federer' , predicting that if that player wins or not.

- **Model Based On Comparison**

  Here in this model we calculated the win percentage of every players with respect to each surface type and stored those percentage values in some matrices.

  Now comparing those win percentage of each player we got our final prediction. This model includes the following steps:

  i. n= total no. of players
  ii. m= total type of surface
  iii. now consider arr, win, per are 3 matrices of dimension nxm
  iv. arr holds total no. of match played by each player on each surface.
  v. win holds total no. of match won by each player on each surface.
  vi. Pre=arr/win , i.e. it holds y=the win percentage of each player on each surface.
  vii. Now for predicting the result we need three attribute player opponent & surface.
  viii. Then we fetched the win percentage of individual player and stored it(suppose in a) and similarly for the opponent in b.
  ix. Now if a is greater than b then we assign ‘1’ to win otherwise we assign ‘0’ to win.

EXPERIMENTAL RESULT

MODEL	LOGISTIC REGRESSION	K-NN	NAIVE BAYES	DESCION TREE
HISTORICAL DATA	62%	59%	60%	55%
IN-GAME DATA	89%	85%	86%	83%

CONFUSION MATRIX

HISTORICAL DATA→

For K-NN : $\begin{matrix} 393 & 262 \\ 284 & 407 \end{matrix}$

For LOGISTIC REGRESSION : $\begin{matrix} 414 & 241 \\ 269 & 422 \end{matrix}$

For NAÏVE-BAYES : $\begin{matrix} 418 & 237 \\ 299 & 392 \end{matrix}$

For DECISION-TREE : $\begin{matrix} 375 & 280 \\ 313 & 378 \end{matrix}$

- **IN-GAME DATA→**

For K-NN : $$\begin{matrix} 97 & 2 \\ 15 & 0 \end{matrix}$$

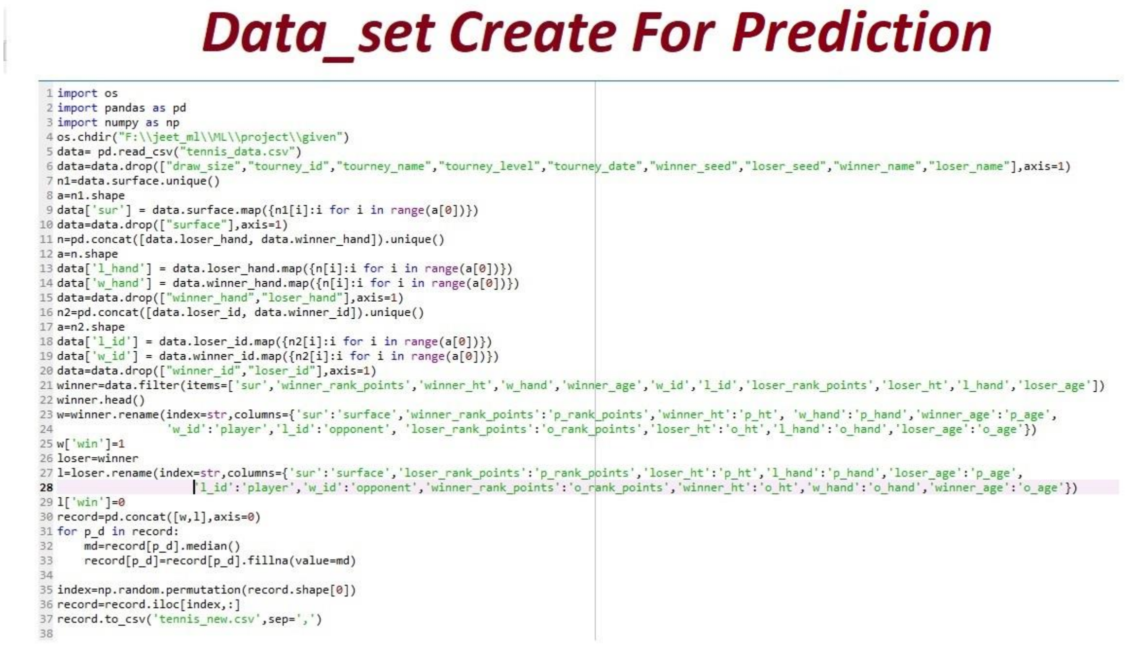
For LOGISTIC REGRESSION : $$\begin{matrix} 99 & 0 \\ 12 & 3 \end{matrix}$$

For NAÏVE-BAYES : $$\begin{matrix} 97 & 7 \\ 13 & 2 \end{matrix}$$

For DECISION-TREE : $$\begin{matrix} 92 & 7 \\ 12 & 3 \end{matrix}$$

Screenshots:

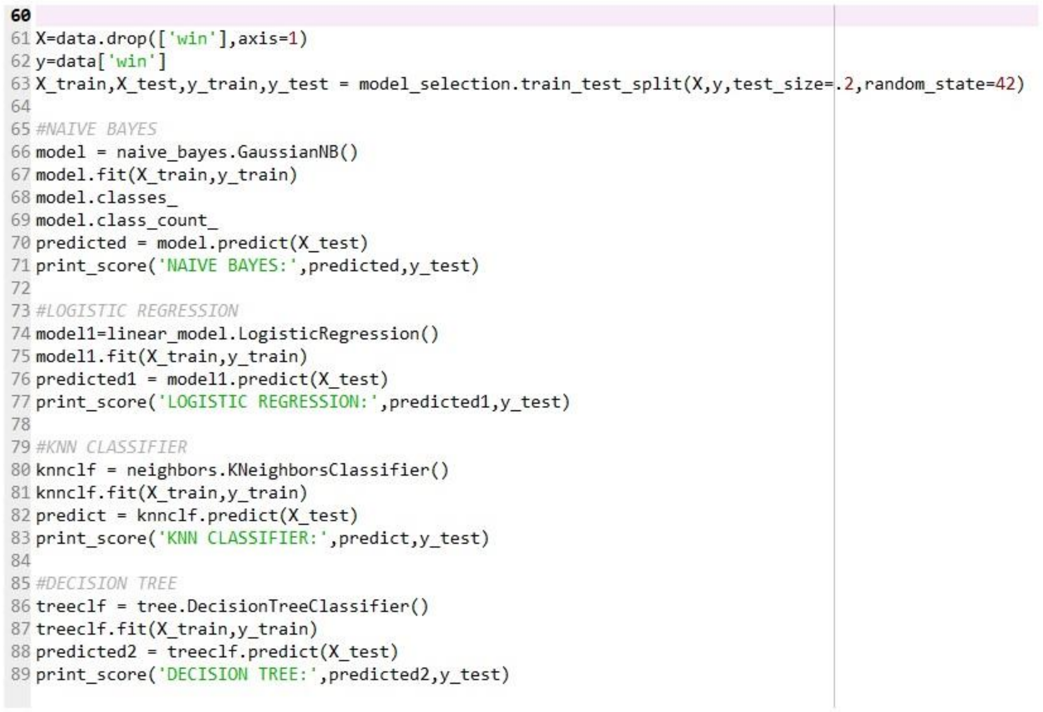
FOR HISTORICAL DATA→



Data_set Create For Prediction

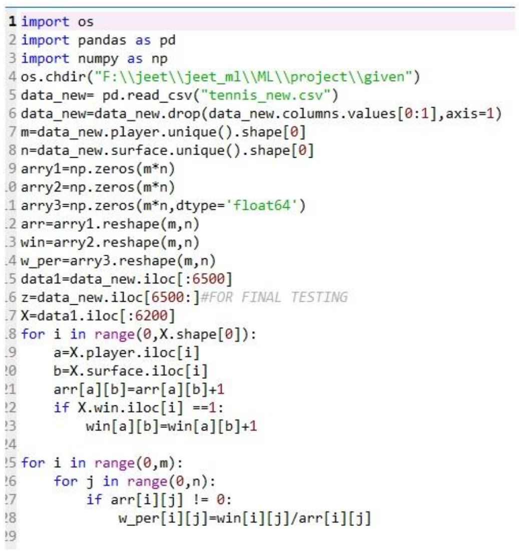
```python
import os
import pandas as pd
import numpy as np
os.chdir("F:\\jeet_ml\\ML\\project\\given")
data= pd.read_csv("tennis_data.csv")
data=data.drop(["draw_size","tourney_id","tourney_name","tourney_level","tourney_date","winner_seed","loser_seed","winner_name","loser_name"],axis=1)
n1=data.surface.unique()
a=n1.shape
data['sur'] = data.surface.map({n1[i]:i for i in range(a[0])})
data=data.drop(["surface"],axis=1)
n=pd.concat([data.loser_hand, data.winner_hand]).unique()
a=n.shape
data['l_hand'] = data.loser_hand.map({n[i]:i for i in range(a[0])})
data['w_hand'] = data.winner_hand.map({n[i]:i for i in range(a[0])})
data=data.drop(["winner_hand","loser_hand"],axis=1)
n2=pd.concat([data.loser_id, data.winner_id]).unique()
a=n2.shape
data['l_id'] = data.loser_id.map({n2[i]:i for i in range(a[0])})
data['w_id'] = data.winner_id.map({n2[i]:i for i in range(a[0])})
data=data.drop(["winner_id","loser_id"],axis=1)
winner=data.filter(items=['sur','winner_rank_points','winner_ht','w_hand','winner_age','w_id','l_id','loser_rank_points','loser_ht','l_hand','loser_age'])
winner.head()
w=winner.rename(index=str,columns={'sur':'surface','winner_rank_points':'p_rank_points','winner_ht':'p_ht', 'w_hand':'p_hand','winner_age':'p_age',
                'w_id':'player','l_id':'opponent', 'loser_rank_points':'o_rank_points','loser_ht':'o_ht','l_hand':'o_hand','loser_age':'o_age'})
w['win']=1
loser=winner
l=loser.rename(index=str,columns={'sur':'surface','loser_rank_points':'p_rank_points','loser_ht':'p_ht','l_hand':'p_hand','loser_age':'p_age',
                'l_id':'player','w_id':'opponent','winner_rank_points':'o_rank_points','winner_ht':'o_ht','w_hand':'o_hand','winner_age':'o_age'})
l['win']=0
record=pd.concat([w,l],axis=0)
for p_d in record:
    md=record[p_d].median()
    record[p_d]=record[p_d].fillna(value=md)

index=np.random.permutation(record.shape[0])
record=record.iloc[index,:]
record.to_csv('tennis_new.csv',sep=',')
```

Prediction By Models

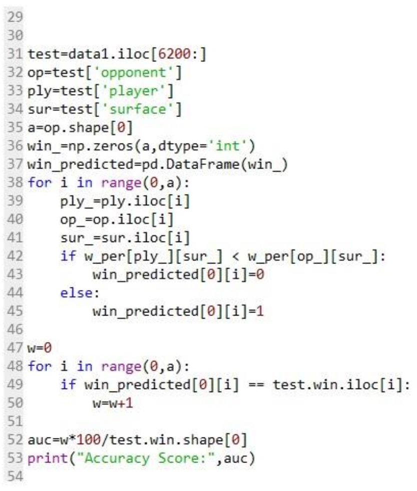


```python

X=data.drop(['win'],axis=1)
y=data['win']
X_train,X_test,y_train,y_test = model_selection.train_test_split(X,y,test_size=.2,random_state=42)

#NAIVE BAYES
model = naive_bayes.GaussianNB()
model.fit(X_train,y_train)
model.classes_
model.class_count_
predicted = model.predict(X_test)
print_score('NAIVE BAYES:',predicted,y_test)

#LOGISTIC REGRESSION
model1=linear_model.LogisticRegression()
model1.fit(X_train,y_train)
predicted1 = model1.predict(X_test)
print_score('LOGISTIC REGRESSION:',predicted1,y_test)

#KNN CLASSIFIER
knnclf = neighbors.KNeighborsClassifier()
knnclf.fit(X_train,y_train)
predict = knnclf.predict(X_test)
print_score('KNN CLASSIFIER:',predict,y_test)

#DECISION TREE
treeclf = tree.DecisionTreeClassifier()
treeclf.fit(X_train,y_train)
predicted2 = treeclf.predict(X_test)
print_score('DECISION TREE:',predicted2,y_test)
```

Comparison Model



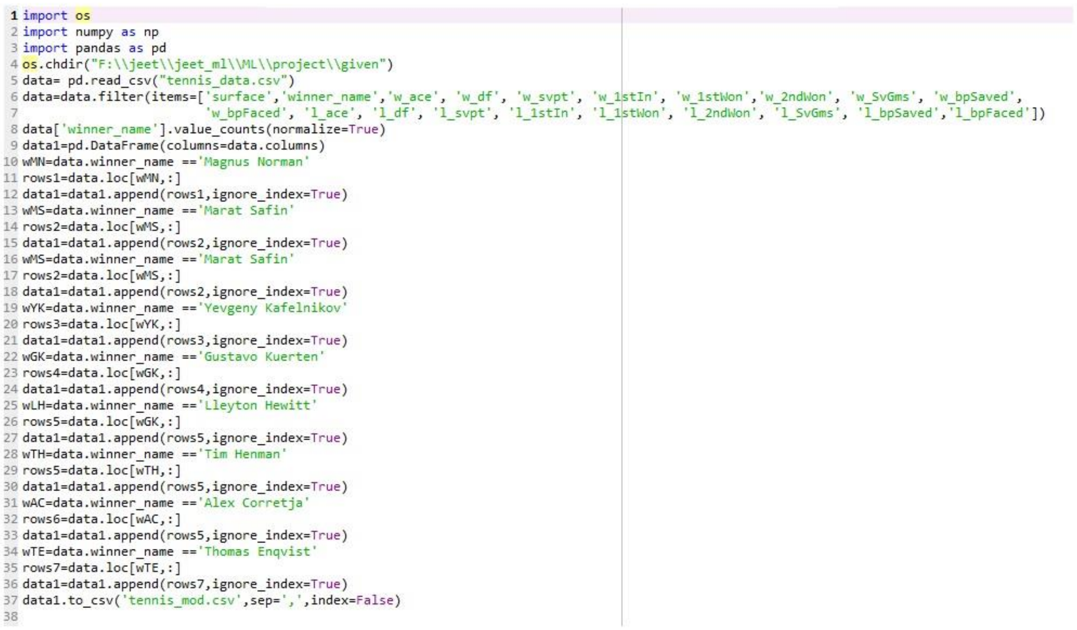
```python
import os
import pandas as pd
import numpy as np
os.chdir("F:\\jeet\\jeet_ml\\ML\\project\\given")
data_new= pd.read_csv("tennis_new.csv")
data_new=data_new.drop(data_new.columns.values[0:1],axis=1)
m=data_new.player.unique().shape[0]
n=data_new.surface.unique().shape[0]
arry1=np.zeros(m*n)
arry2=np.zeros(m*n)
arry3=np.zeros(m*n,dtype='float64')
arr=arry1.reshape(m,n)
win=arry2.reshape(m,n)
w_per=arry3.reshape(m,n)
data1=data_new.iloc[:6500]
z=data_new.iloc[6500:]#FOR FINAL TESTING
X=data1.iloc[:6200]
for i in range(0,X.shape[0]):
    a=X.player.iloc[i]
    b=X.surface.iloc[i]
    arr[a][b]=arr[a][b]+1
    if X.win.iloc[i] ==1:
        win[a][b]=win[a][b]+1

for i in range(0,m):
    for j in range(0,n):
        if arr[i][j] != 0:
            w_per[i][j]=win[i][j]/arr[i][j]

```



```python


test=data1.iloc[6200:]
op=test['opponent']
ply=test['player']
sur=test['surface']
a=op.shape[0]
win_=np.zeros(a,dtype='int')
win_predicted=pd.DataFrame(win_)
for i in range(0,a):
    ply_=ply.iloc[i]
    op_=op.iloc[i]
    sur_=sur.iloc[i]
    if w_per[ply_][sur_] < w_per[op_][sur_]:
        win_predicted[0][i]=0
    else:
        win_predicted[0][i]=1

w=0
for i in range(0,a):
    if win_predicted[0][i] == test.win.iloc[i]:
        w=w+1

auc=w*100/test.win.shape[0]
print("Accuracy Score:",auc)

```

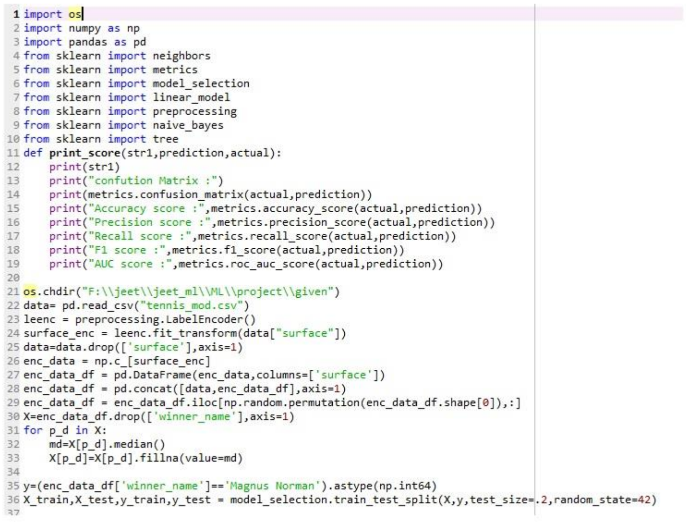
➢ **IN-GAME DATA→**

Selection Of Player

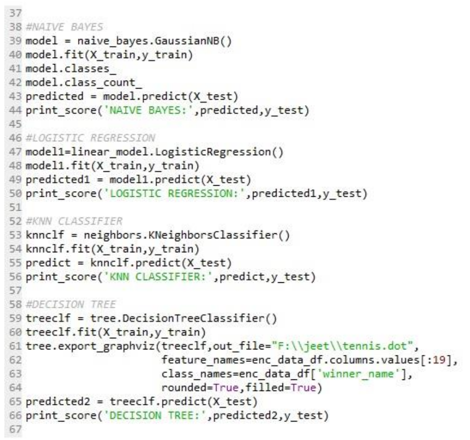


```python
import os
import numpy as np
import pandas as pd
os.chdir("F:\\jeet\\jeet_ml\\ML\\project\\given")
data= pd.read_csv("tennis_data.csv")
data=data.filter(items=['surface','winner_name','w_ace', 'w_df', 'w_svpt', 'w_1stIn', 'w_1stWon','w_2ndWon', 'w_SvGms', 'w_bpSaved',
                        'w_bpFaced', 'l_ace', 'l_df', 'l_svpt', 'l_1stIn', 'l_1stWon', 'l_2ndWon', 'l_SvGms', 'l_bpSaved','l_bpFaced'])
data['winner_name'].value_counts(normalize=True)
data1=pd.DataFrame(columns=data.columns)
wMN=data.winner_name =='Magnus Norman'
rows1=data.loc[wMN,:]
data1=data1.append(rows1,ignore_index=True)
wMS=data.winner_name =='Marat Safin'
rows2=data.loc[wMS,:]
data1=data1.append(rows2,ignore_index=True)
wMS=data.winner_name =='Marat Safin'
rows2=data.loc[wMS,:]
data1=data1.append(rows2,ignore_index=True)
wYK=data.winner_name =='Yevgeny Kafelnikov'
rows3=data.loc[wYK,:]
data1=data1.append(rows3,ignore_index=True)
wGK=data.winner_name =='Gustavo Kuerten'
rows4=data.loc[wGK,:]
data1=data1.append(rows4,ignore_index=True)
wLH=data.winner_name =='Lleyton Hewitt'
rows5=data.loc[wGK,:]
data1=data1.append(rows5,ignore_index=True)
wTH=data.winner_name =='Tim Henman'
rows5=data.loc[wTH,:]
data1=data1.append(rows5,ignore_index=True)
wAC=data.winner_name =='Alex Corretja'
rows6=data.loc[wAC,:]
data1=data1.append(rows5,ignore_index=True)
wTE=data.winner_name =='Thomas Enqvist'
rows7=data.loc[wTE,:]
data1=data1.append(rows7,ignore_index=True)
data1.to_csv('tennis_mod.csv',sep=',',index=False)

```

Prediction By Models



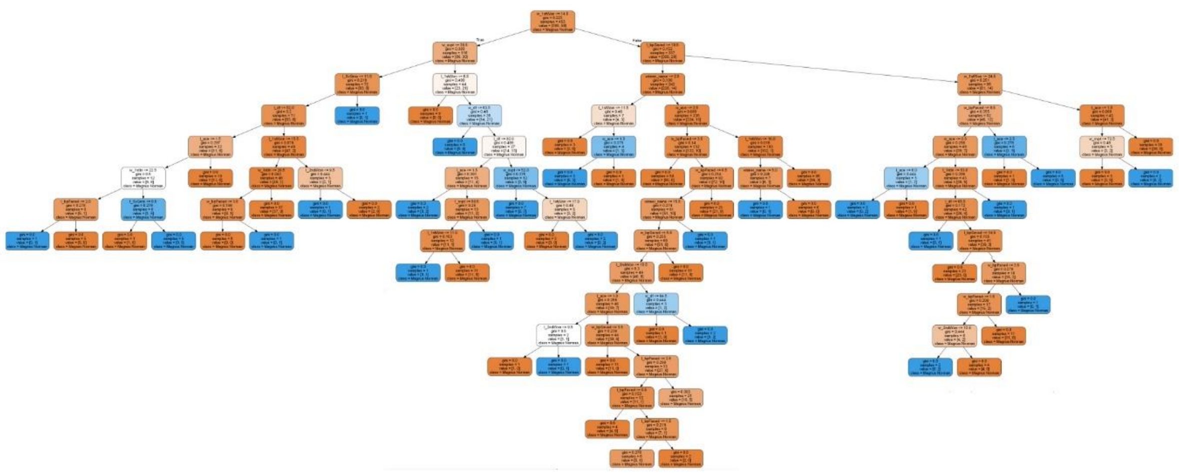
```python
import os
import numpy as np
import pandas as pd
from sklearn import neighbors
from sklearn import metrics
from sklearn import model_selection
from sklearn import linear_model
from sklearn import preprocessing
from sklearn import naive_bayes
from sklearn import tree
def print_score(str1,prediction,actual):
    print(str1)
    print("confution Matrix :")
    print(metrics.confusion_matrix(actual,prediction))
    print("Accuracy score :",metrics.accuracy_score(actual,prediction))
    print("Precision score :",metrics.precision_score(actual,prediction))
    print("Recall score :",metrics.recall_score(actual,prediction))
    print("F1 score :",metrics.f1_score(actual,prediction))
    print("AUC score :",metrics.roc_auc_score(actual,prediction))

os.chdir("F:\\jeet\\jeet_ml\\ML\\project\\given")
data= pd.read_csv("tennis_mod.csv")
leenc = preprocessing.LabelEncoder()
surface_enc = leenc.fit_transform(data["surface"])
data=data.drop(['surface'],axis=1)
enc_data = np.c_[surface_enc]
enc_data_df = pd.DataFrame(enc_data,columns=['surface'])
enc_data_df = pd.concat([data,enc_data_df],axis=1)
enc_data_df = enc_data_df.iloc[np.random.permutation(enc_data_df.shape[0]),:]
X=enc_data_df.drop(['winner_name'],axis=1)
for p_d in X:
    md=X[p_d].median()
    X[p_d]=X[p_d].fillna(value=md)

y=(enc_data_df['winner_name']=='Magnus Norman').astype(np.int64)
X_train,X_test,y_train,y_test = model_selection.train_test_split(X,y,test_size=.2,random_state=42)
```



```python

#NAIVE BAYES
model = naive_bayes.GaussianNB()
model.fit(X_train,y_train)
model.classes_
model.class_count_
predicted = model.predict(X_test)
print_score('NAIVE BAYES:',predicted,y_test)

#LOGISTIC REGRESSION
model1=linear_model.LogisticRegression()
model1.fit(X_train,y_train)
predicted1 = model1.predict(X_test)
print_score('LOGISTIC REGRESSION:',predicted1,y_test)

#KNN CLASSIFIER
knnclf = neighbors.KNeighborsClassifier()
knnclf.fit(X_train,y_train)
predict = knnclf.predict(X_test)
print_score('KNN CLASSIFIER:',predict,y_test)

#DECISION TREE
treeclf = tree.DecisionTreeClassifier()
treeclf.fit(X_train,y_train)
tree.export_graphviz(treeclf,out_file="F:\\jeet\\tennis.dot",
                     feature_names=enc_data_df.columns.values[:19],
                     class_names=enc_data_df['winner_name'],
                     rounded=True,filled=True)
predicted2 = treeclf.predict(X_test)
print_score('DECISION TREE:',predicted2,y_test)

```

DECITION-TREE->

Decition Tree Diagram



Future Scope of Improvements

Future work includes playing with the feature representation, as well as experimenting with different algorithms. Possible other approaches that could produce better results are using only the top tournaments for training data, experimenting with more kernels and larger ranges of parameters, as well as using data from farther in the past. Given the good performance of our baselines, it may also be feasible to actually use a boosting algorithm using our baselines as the weak learners to obtain better results.

Using this data set we can predict other attributes also. Like the stats of a player. The data here we get is actually not so large. It only consist the information of the tennis matches played in one single year. We know it is not so easy to predict a the match result before the match start. But although we get the accuracy near about 60%. If the data is more large and hold the records of a long time period then I am sure that our model we able to predict more than 75%.

By using this model we can easily bet a tennis match. We can see that the chance of our winning is 55%-60%.

There is hardly one match between two players. So for getting the higher result we need more data.

Certificate

This is to certify that Mr. Jeet Saha of Academy Of Technology, registration number: 151690110145 has successfully completed a project on Tennis Match Prediction using ML (Machine Learning Models) under the guidance of Mr. Titas Roychowdhury

----

[*Titas Roychowdhury*]

Globsyn Finishing School

Certificate

This is to certify that Mr.Sanmitra Kumar of Academy Of Technology, registration number: 151690110165 has successfully project on Tennis Match Prediction using ML (Machine Learning Models) under the guidance of Mr. Titas Roychowdhury

----

[*Titas Roychowdhury*]

Globsyn Finishing School

Certificate

This is to certify that Mr.Arijit Choudhury of Academy Of Technology, registration number: 151690110018 has successfully project on Tennis Match Prediction using ML (Machine Learning Models) under the guidance of Mr. Titas Roychowdhury

----

[*Titas Roychowdhury*]

Globsyn Finishing School

Certificate

This is to certify that Miss.Sushmita Shaw of Academy Of Technology, registration number: 151690110187 has successfully project on Tennis Match Prediction using ML (Machine Learning Models) under the guidance of Mr. Titas Roychowdhury

----

[*Titas Roychowdhury*]

Globsyn Finishing School