Predicting the Winner of a Tennis Match using Machine Learning Techniques

MSc Research Project
Data Analytics

Akshaya Sekar
Student ID: x18138977

School of Computing
National College of Ireland

Supervisor: Dr Dondio Pierpaolo

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Akshaya Sekar
Student ID:	x18138977
Programme:	Data Analytics
Year:	2018
Module:	MSc Research Project
Supervisor:	Dr Dondio Pierpaolo
Submission Due Date:	20/12/2018
Project Title:	Predicting the Winner of a Tennis Match using Machine Learning Techniques
Word Count:	672
Page Count:	10

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	3rd February 2020

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Predicting the Winner of a Tennis Match using Machine Learning Techniques

Akshaya Sekar
x18138977

1 Introduction

The configuration manual represents every step of the implementation process in a detailed manner.The hardware and software specifications are mentioned for the research project on the topic," Predicting the winner of a tennis match using machine learning techniques".The goal of this project is to predict the winner of the match with the individual player statistics using various machine learning models such as SVM, Logistic regression, Random forest, Naive Bayes.PCA was used for dimensionality reduction and random search Hyper parameter tuning was performed to increase the efficiency of the models.

2 System Specification

This project was implemented on the cloud platform Google colaboratory also known as Colab. The colab supports GPU and TPU.Bisong (2019)

2.1 Hardware

- Google Colab: 2vCPU @ 2.2GHz
- The GPU Instance was 250GB
- The RAM was 13 GB
- The Disk Space was 32GB

2.2 Software

Python programming language was used to implement the project. The entire pre-processing tasks such as cleaning, encoding, dimension reduction implementation and evaluation was performed in Python..

3 Importing Libraries

Some libraries required are pre-defined in the cloud platform. The other necessary libraries were imported whenever required. This step involves importing the required libraries.

```
[ ]
    #libraries
    import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
    import seaborn as sns
    import pandas_profiling
    from sklearn.preprocessing import LabelEncoder
    from sklearn.decomposition import PCA
    from sklearn.model_selection import train_test_split
    from sklearn.svm import SVC
    from sklearn.metrics import auc, roc_curve
    from sklearn.metrics import accuracy_score
    from sklearn.metrics import f1_score
    from sklearn.metrics import cohen_kappa_score
    from sklearn.naive_bayes import GaussianNB
    from sklearn.model_selection import RandomizedSearchCV
    %matplotlib inline
```

Figure 1: Importing Libraries

4 Data Extraction

4.1 Importing Files

In this step, the data set is mounted in the google drive and then the file is imported from the google drive.

```
[ ] #from google.colab import files
    #uploaded = files.upload()
    from google.colab import drive
    drive.mount('/content/drive')
    %cd /content/drive/My Drive/Colab Notebooks/

    Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&r

    Enter your authorization code:
    ··········
    Mounted at /content/drive
    /content/drive/My Drive/Colab Notebooks
```

Figure 2: Importing Data

4.2 Set Path

In this step, the path of the data set is given and the data is read.

```
[ ] #import io
    #raw_data = pd.read_csv(io.BytesIO(uploaded['clinvar_conflicting.csv']))
    path='/content/drive/My Drive/Colab Notebooks/Dataset/'
```

Figure 3: Working directory path

4.3 Reading the data

```
#importing data
raw = pd.read_csv(path+'Stats (1).csv')
```

Figure 4: Reading Data

5 Exploratory Data Analysis (EDA)

The Exploratory Data Analysis was done with the help of pandas profiling in Python. The pandas profiling is a one line code which gives a better understanding about the insights of the data . It analysis the data and gives a HTML format report of all the missing values, outliers, class balance, correlations and other basic details about the dataset etc.

```
pandas_profiling.ProfileReport(raw)
```

Number of variables	22
Number of observations	20240
Total Missing (%)	6.1%
Total size in memory	3.3 MiB
Average record size in memory	169.0 B

riables types

Numeric	17
Categorical	2
Boolean	1
Date	0
Text (Unique)	0
Rejected	2
Unsupported	0

Figure 5: Exploratory Data Analysis

5.1 Removing null values

In this process, the null values are removed from the raw dataset.This will increase the quality of the dataset and helps to give efficient result.

```
#removing NAs
raw = raw.dropna()
```

Figure 6: Removing null values

5.2 Checking Class Imbalance

The class should be equally balanced to get efficient results, hence the process of under sampling or over sampling takes place depending on the data. Here in this dataset, the class was equally balanced.

```
#ploting class balance
plt.figure(figsize=(8, 8))
sns.countplot('winner', data=raw)
plt.title('Class')
plt.show()
```

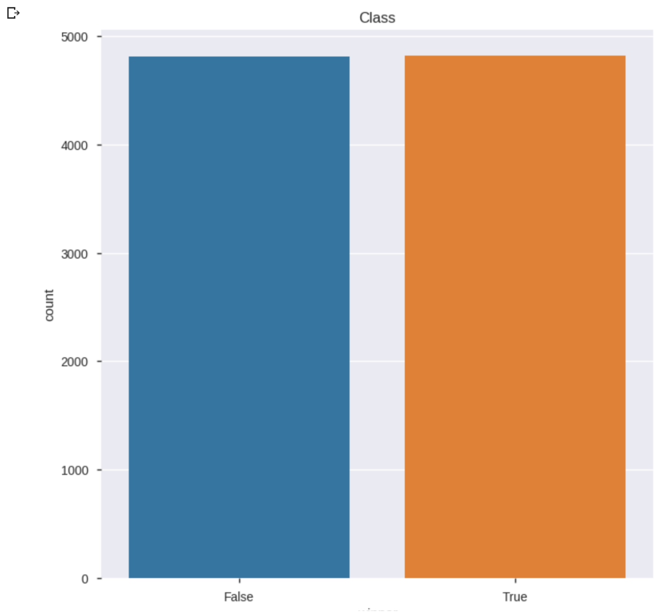

Figure 7: Class Imbalance check

5.3 Dropping the unwanted columns

The columns which are irrelevant and columns with special characters are removed.

```
#droping unwanted coulmns
raw = raw.drop(["match_id", "player_id"],axis = 1)
```

Figure 8: Removing unwanted columns

6 Data Pre-processing

6.1 Dependent and Independent variables

In this section, we are splitting the data set in to dependent and independent variables. Here, X is denoted as the independent variable and y is denoted as the dependent varibale.

```
[ ] #X- indipendent variables, y- dependent variable
    X = raw.iloc[:, :-1].values
    y = raw.iloc[:, -1].values
```

Figure 9: Dependent and independent variables

6.2 Encoding the data

Since the machine learning models cannot accept characters, we will encode the dependent variables as 0's and 1's. This process is known as Label encoding. The Target column is label encoded as 0's for loser and 1's for winner.

```
[ ] #lable encoding for catagorical data
    bin_cols = raw.nunique()[raw.nunique() == 2].keys().tolist()
    le = LabelEncoder()
    for i in bin_cols :
        raw[i] = le.fit_transform(raw[i])
```

Figure 10: Label Encoding

7 Dimensionality Reduction

In this project, the Principle Component Analysis is used for dimensionality reduction. Since the dataset has continuous values, PCA is used for dimensionality reduction. PCA helps to reduce the number of columns by having a summary of all the important features with high variance. This helps to increase the efficiency of the models and reduce the computation time.

```
#PCA for dimensonality reduction
pca = PCA(n_components=2)
X = pca.fit_transform(X)

[ ] pca.explained_variance_ratio_

array([0.94146386, 0.05729056])
```

Figure 11: Dimensionality Reduction

8 Training and testing dataset

In this stage, the data set was split in to training and testing in 80:20 ratio.

```
[ ] # Splitting the dataset into the Training set and Test set
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20)
```

Figure 12: Training and Testing

9 Machine learning models

There are four machine learning models implemented in this project. SVM, Naive Bayes, Logistic Regression, Random Forest.

9.1 Support Vector Machine

```
[ ] #implementing SVC
    svm = SVC()
    svm.fit(X_train, y_train)
    y_pred = svm.predict(X_test)
    acc = accuracy_score(y_pred,y_test)
    print("Accuracy :", acc)
    false_positive_rate, true_positive_rate, thresholds = roc_curve(y_test, y_pred)
    score_auc = auc(false_positive_rate, true_positive_rate)
    print("AUC Score =", score_auc)
    score_f1 = f1_score(y_test, y_pred)
    print("F1 Score :", score_f1)
    score_kappa = cohen_kappa_score(y_pred,y_test)
    print("Cohens Kappa :", score_kappa)
    print("AUC-ROC Curve: ")
    plt.plot([0, 1], [0, 1], linestyle='--')
    plt.plot(false_positive_rate, true_positive_rate,marker='.')
    plt.show()
```

Figure 13: SVM

```
/usr/local/lib/python3.6/dist-packages/sklearn/svm/base.py:193: FutureWarning: The default value of gamma will change from 'auto' to 'scale' in version 0.2
  "avoid this warning.", FutureWarning)
Accuracy : 0.7197716658017644
AUC Score = 0.7197509280145122
F1 Score : 0.7169811320754718
Cohens Kappa : 0.4395184268726079
AUC-ROC Curve:
```

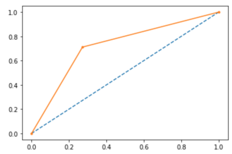

Figure 14: Support Vector Machine Result and AUC curve

9.1.1 Hyper parameter tuning for SVM

The hyper parameter tuning helps in choosing the best parameters which increases the efficiency of the models. Here, the random search Hyper parameter tuning is used.

```
[ ] #seting hyper

    smc_params = {'C': range(1, 10, 1), 'gamma': np.arange(0.1, 1, 0.1), 'kernel': ['rbf', 'poly']}
```

```
[ ] #random search cv for hyper paramter tuning
    random_search = RandomizedSearchCV(estimator = svm, param_distributions = smc_params, n_iter = 10, cv = 5, verbose=2, random_state=42, n_jobs = -1)
    random_search.fit(X_train, y_train)

    Fitting 5 folds for each of 10 candidates, totalling 50 fits
    [Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
```

```
[ ] #checking the best hyper paramters
    print('\n Best estimator:')
    print(random_search.best_estimator_)
    print('\n Best hyperparameters:')
    print(random_search.best_params_)
```

Figure 15: Selecting the parameters

9.2 Naive Bayes

This Gaussian Naive Bayes is split in to training and testing and then evaluated in terms of Accuracy, Auc, F1 score and Kappa.

```
[ ] #implementing Gaussian naive bayes
    gaussian = GaussianNB()
    gaussian.fit(X_train, y_train)
    # Predicting the Test set results
    y_pred = gaussian.predict(X_test)
    acc = accuracy_score(y_pred,y_test)
    print("Accuracy :", acc)
    false_positive_rate, true_positive_rate, thresholds = roc_curve(y_test, y_pred)
    score_auc = auc(false_positive_rate, true_positive_rate)
    print("AUC Score =", score_auc)
    score_f1 = f1_score(y_test, y_pred)
    print("F1 Score :", score_f1)
    score_kappa = cohen_kappa_score(y_pred,y_test)
    print("Cohens Kappa :", score_kappa)
    print("AUC-ROC Curve: ")
    plt.plot([0, 1], [0, 1], linestyle='--')
    plt.plot(false_positive_rate, true_positive_rate,marker='.')
    plt.show()
```

Figure 16: Naive Bayes

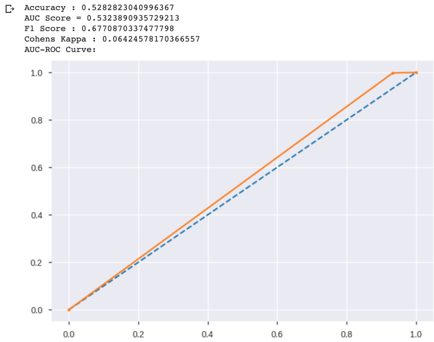

Figure 17: Naive Bayes Result and AUC curve

9.3 Random Forest

Here, the Random Forest data is split in to training and testing and evaluated in terms of Accuracy, F1 score, Auc and Cohens kappa. It is seen that Random Forest has the highest accuracy of 68%

```
[ ] #implementing random forest
    from sklearn.ensemble import RandomForestClassifier
    randomf = RandomForestClassifier(n_estimators = 690, min_samples_split = 6, min_samples_leaf = 1, max_features ='auto', max_depth=10)
    randomf.fit(X_train, y_train)
    # Predicting the Test set results
    y_pred = randomf.predict(X_test)
    acc = accuracy_score(y_pred,y_test)
    print("Accuracy :", acc)
    false_positive_rate, true_positive_rate, thresholds = roc_curve(y_test, y_pred)
    score_auc = auc(false_positive_rate, true_positive_rate)
    print("AUC Score =", score_auc)
    score_f1 = f1_score(y_test, y_pred)
    print("F1 Score :", score_f1)
    score_kappa = cohen_kappa_score(y_pred,y_test)
    print("Cohens Kappa :", score_kappa)
    print("AUC-ROC Curve: ")
    plt.plot([0, 1], [0, 1], linestyle='--')
    plt.plot(false_positive_rate, true_positive_rate,marker='.')
    plt.show()
```

Figure 18: Random Forest

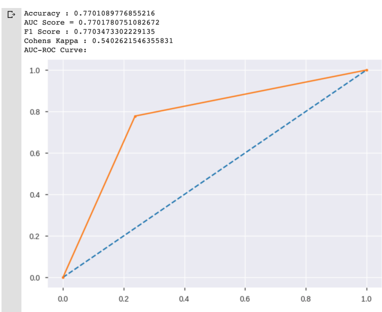

Figure 19: Random forest result and AUC curve

9.3.1 Random Forest Hyper parameter tuning

The various parameters for hyper parameter tuning are set and the best parameters were selected using the random search hyper parameter tuning.

```
[ ] #seting range for hyper parameter
    randomf_params = {'n_estimators': range(200, 2000, 10),
                      'max_features': ['auto', 'sqrt'],
                      'max_depth': range(10, 110, 10),
                      'min_samples_split': range(2, 10, 1),
                      'min_samples_leaf': [1, 2, 4]}
```

```
[ ] m search cv for hyper paramter tuning
    _search = RandomizedSearchCV(estimator = randomf, param_distributions = randomf_params, n_iter = 50, cv = 10, verbose=2, random_state=42, n_jobs = -1)
    _search.fit(X_train, y_train)
```

```
Fitting 10 folds for each of 50 candidates, totalling 500 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
/usr/local/lib/python3.6/dist-packages/joblib/externals/loky/process_executor.py:706: UserWarning: A worker stopped while some jobs were given to the execu
  "timeout or by a memory leak.", UserWarning
[Parallel(n_jobs=-1)]: Done  37 tasks      | elapsed:  6.5min
[Parallel(n_jobs=-1)]: Done 158 tasks      | elapsed: 20.5min
```

```
[ ] print('\n Best estimator:')
    print(random_search.best_estimator_)
    print('\n Best hyperparameters:')
    print(random_search.best_params_)
```

Figure 20: Random search Hyper Parameter tuning of RF

9.4 Logistic Regression

The Logistic Regression Evaluation is shown below. After splitting the training and testing data set, the accuracy, F1 score, AUC and Kappa are evaluated.

```
[ ] #implementing logistic regression
    from sklearn.linear_model import LogisticRegression
    logi = LogisticRegression(penalty = 'l1', C = 1.0)
    logi. Loading... n, y_train)
    y_pred = logi.predict(X_test)
    acc = accuracy_score(y_pred,y_test)
    print("Accuracy :", acc)
    false_positive_rate, true_positive_rate, thresholds = roc_curve(y_test, y_pred)
    score_auc = auc(false_positive_rate, true_positive_rate)
    print("AUC Score =", score_auc)
    score_f1 = f1_score(y_test, y_pred)
    print("F1 Score :", score_f1)
    score_kappa = cohen_kappa_score(y_pred,y_test)
    print("Cohens Kappa :", score_kappa)
    print("AUC-ROC Curve: ")
    plt.plot([0, 1], [0, 1], linestyle='--')
    plt.plot(false_positive_rate, true_positive_rate,marker='.')
    plt.show()
```

Figure 21: Logistic Regression

```
[ ] /usr/local/lib/python3.6/dist-packages/sklearn/linear_model/logistic.py:432: FutureWarning: Default solver will be changed to 'lbfgs'
      FutureWarning)
    Accuracy : 0.7540217955371044
    AUC Score = 0.7543136785999746
    F1 Score : 0.7785046728971963
    Cohens Kappa : 0.5083289283441943
    AUC-ROC Curve:
```

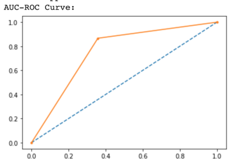

Figure 22: Logistic Regression result and AUC curve

9.4.1 Hyper parameter tuning for Logistic Regression

The random search hyper parameter tuning is done with various parameters to select the best parameters.

```
[ ] #seting range for hyper pramter tuning
    logi_param={"C":np.logspace(-3,3,7), "penalty":["l1","l2"],}

    #random search cv for hyper paramter tuning
    random_search = RandomizedSearchCV(logi, param_distributions=logi_param, n_iter=50, scoring='accuracy', n_jobs= -1, verbose=3)
```

```
[ ] random_search.fit(X_train, y_train)
```

```
Fitting 3 folds for each of 14 candidates, totalling 42 fits
/usr/local/lib/python3.6/dist-packages/sklearn/model_selection/_split.py:1978: FutureWarning: The default value of cv will change
  warnings.warn(CV_WARNING, FutureWarning)
/usr/local/lib/python3.6/dist-packages/sklearn/model_selection/_search.py:266: UserWarning: The total space of parameters 14 is sm
  % (grid_size, self.n_iter, grid_size), UserWarning)
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done  42 out of  42 | elapsed:    2.2s finished
/usr/local/lib/python3.6/dist-packages/sklearn/linear_model/logistic.py:432: FutureWarning: Default solver will be changed to 'lbf
  FutureWarning)
RandomizedSearchCV(cv='warn', error_score='raise-deprecating',
                   estimator=LogisticRegression(C=1.0, class_weight=None,
                                                dual=False, fit_intercept=True,
                                                intercept_scaling=1,
                                                l1_ratio=None, max_iter=100,
                                                multi_class='warn', n_jobs=None,
                                                penalty='l1', random_state=None,
                                                solver='warn', tol=0.0001,
                                                verbose=0, warm_start=False),
                   iid='warn', n_iter=50, n_jobs=-1,
                   param_distributions={'C': array([1.e-03, 1.e-02, 1.e-01, 1.e+00, 1.e+01, 1.e+02, 1.e+03]),
                                        'penalty': ['l1', 'l2']},
                   pre_dispatch='2*n_jobs', random_state=None, refit=True,
                   return_train_score=False, scoring='accuracy', verbose=3)
```

Figure 23: Logistic Regression Hyper parameter

```
[ ]
    print('\n Best estimator:')
    print(random_search.best_estimator_)
    print('\n Best hyperparameters:')
    print(random_search.best_params_)
```

```
 Best estimator:
LogisticRegression(C=100.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l1',
                   random_state=None, solver='warn', tol=0.0001, verbose=0,
                   warm_start=False)

 Best hyperparameters:
{'penalty': 'l1', 'C': 100.0}
```

Figure 24: Selecting the best parameters

References

Bisong, E. (2019). Google colaboratory, *Building Machine Learning and Deep Learning Models on Google Cloud Platform*, Springer, pp. 59–64.