PUNE INSTITUTE OF COMPUTER TECHNOLOGY
DHANKAWADI, PUNE

DATA MINING AND WAREHOUSING MINI-PROJECT REPORT
ON

“PREDICTING SOCCER GAME RESULTS USING VARIOUS MODELS”

SUBMITTED BY

Nachiket Erlekar	41434
Ashay Koradia	41429

Under the guidance of
Prof. Deepali Kadam

DEPARTMENT OF COMPUTER ENGINEERING

Academic Year 2020-21

Contents

1 Problem Statement

Consider a labeled dataset belonging to an application domain. Apply suitable data preprocessing steps such as handling of null values, data reduction, discretization. For prediction of class labels of given data instances, build classifier models using different techniques (minimum 3), analyze the confusion matrix and compare these models. Also apply cross validation while preparing the training and testing datasets.

2 Abstract

Classification is a form of data analysis that extracts models describing importantdata classes. Such models, called classifiers, predict categorical (discrete, unordered)class labels. For example, we can build a classification model to categorize bank loanapplications as either safe or risky. Such analysis can help provide us with a better understanding of the data at large. In this project we use multiple classification models toanalyse the outcome of Soccer game played between various teams. Use apply suitabledata preprocessing steps.We then compare performance of classification models to findwhich one is the best

3 Hardware and Software Requirements

3.1 Hardware Requirements

1. 500 GB HDD
2. 4GB RAM
3. Monitor
4. Keyboard

3.2 Software Requirements

1. 64 bit Open Source Operating System like Ubuntu 18.04
2. Python 3
3. Google Colab
4. Different Libraries
5. Libararies like sklearn, pandas, matplotlib

4 INTRODUCTION

We have been provided with the data regarding various aspects of the home team, the opposition team and their supporters for a number of Soccer games.
The Data fields are

1. Id – Unique id given to each game.
2. game_seq – Sequence of the game in the history of FIH (International Soccer Federation).
3. season_end – Year in which the corresponding season ended.
4. date – Date on which the game was played.
5. season_game_seq – Sequence of the game in the corresponding season.
6. playoff – Whether the game is a playoffs game.
7. team_id – Unique id for the home team.
8. Elo – Elo rating for the home team before the game.
9. opp_team_id – Unique id for the opposition team.
10. opp_Elo – Elo rating for the opposition team before the game.
11. win_equivalent – Equivalent number of wins for the home team in a season.
12. bet_ratio – Fraction of bets placed on the home team.
13. home_crowd – Number of supporters for the home team.
14. opp_crowd – Number of supporters for the opposition team.
15. total_crowd – Total number of attendees for the game.
16. game_result – Win or loss for the home team (Win - 1, Loss - 0).

The train set contains 45000 records while the test set contains 13107 records.We drop the date column from our analysis. The null entries are as follows

Attribute	Null Count
Elo	9197
opp_Elo	7006
win_equivalent	12263

Table 1: Null Counts

We fill the null Elo and opp_elo entries with the mean value of Elo and opp_Elo attribute respectively i.e. 1501.184 1501.837

5 OBJECTIVE

- To understand data preprocessing
- To perform classification on dataset and predict labels for test dataset.

6 Scope

We select dataset of soccer games of various seasons. We try to apply many models and compare which one is the best model amongst them.

7 System Architecture

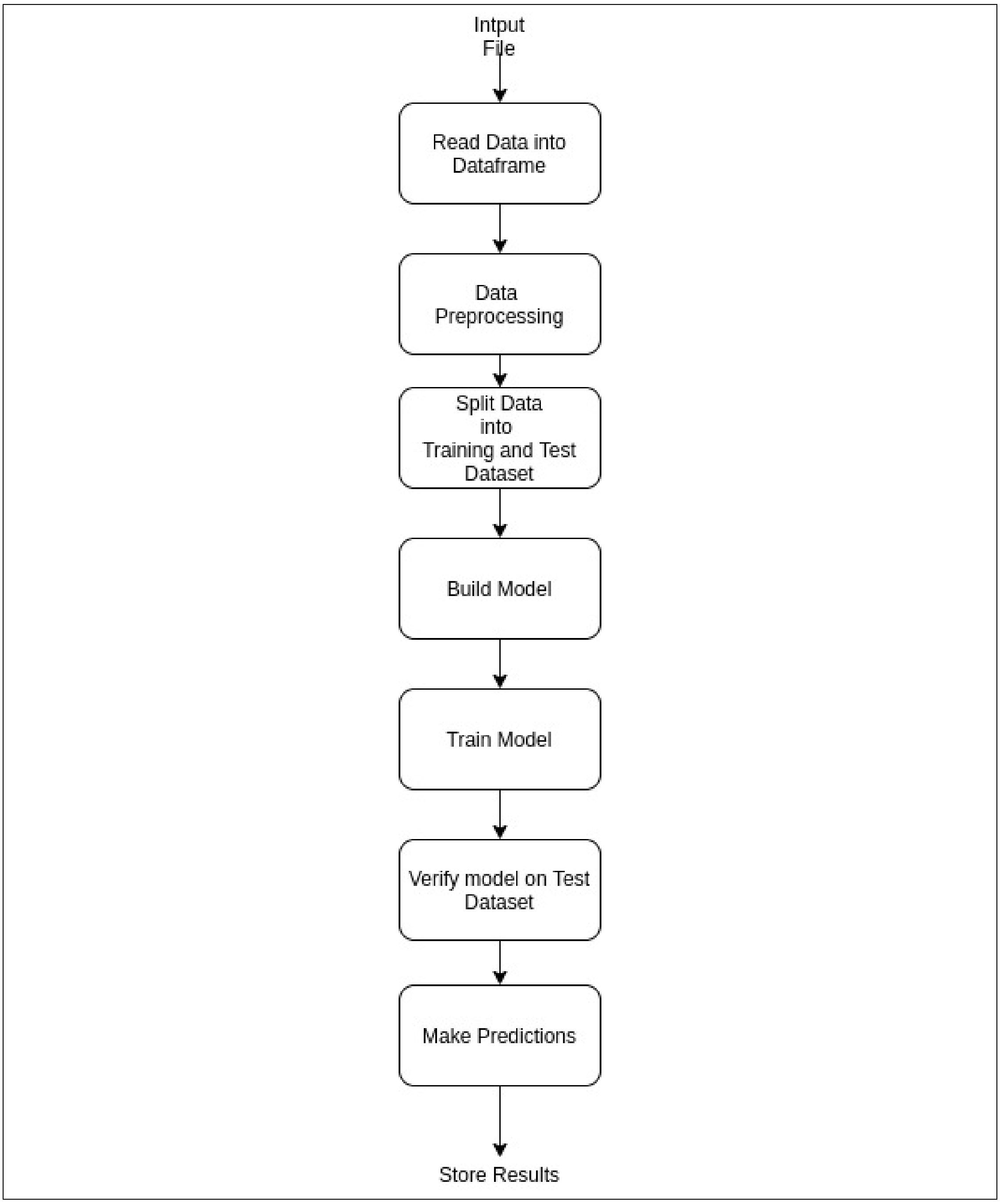
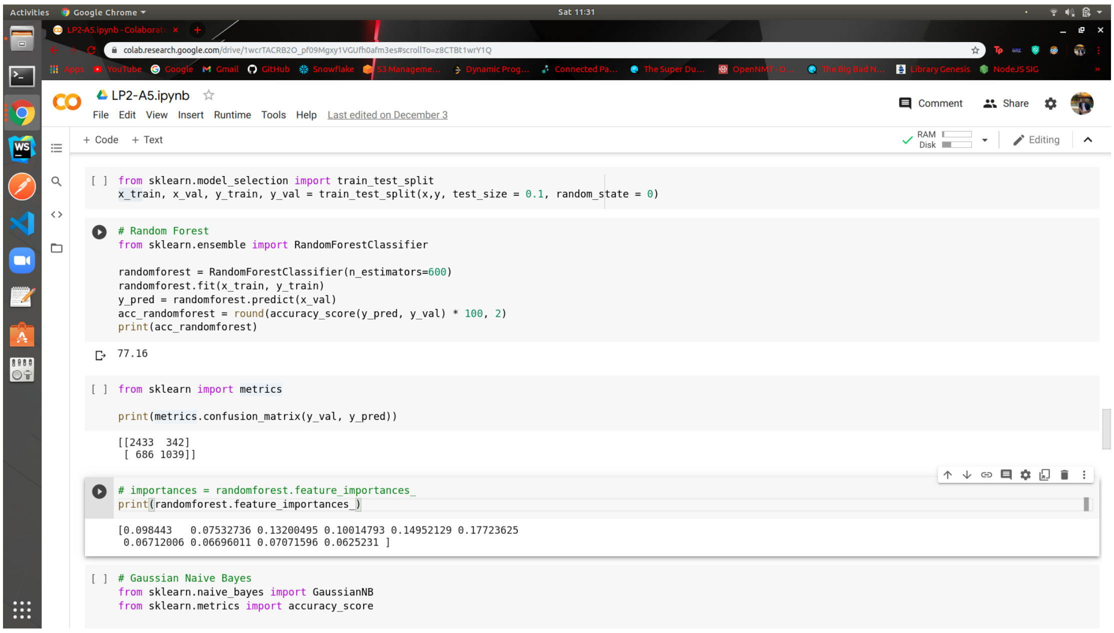


Figure 1: System Architecture

8 Test Cases


LP2-A5.ipynb
File Edit View Insert Runtime Tools Help Last edited on December 3
Comment
Share
+ Code + Text
RAM
Disk
Editing

```
from sklearn.model_selection import train_test_split
x_train, x_val, y_train, y_val = train_test_split(x,y, test_size = 0.1, random_state = 0)
```

```
# Random Forest
from sklearn.ensemble import RandomForestClassifier

randomforest = RandomForestClassifier(n_estimators=600)
randomforest.fit(x_train, y_train)
y_pred = randomforest.predict(x_val)
acc_randomforest = round(accuracy_score(y_pred, y_val) * 100, 2)
print(acc_randomforest)
```

```
77.16
```

```
from sklearn import metrics

print(metrics.confusion_matrix(y_val, y_pred))
```

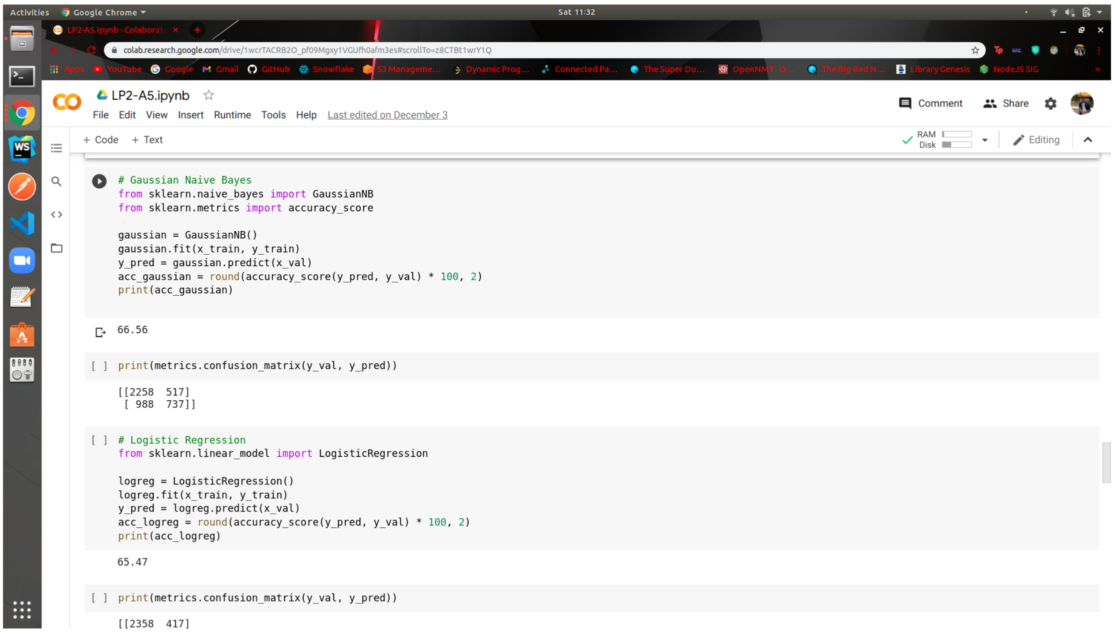
```
[[2433  342]
 [ 686 1039]]
```

```
# importances = randomforest.feature_importances_
print(randomforest.feature_importances_)
```

```
[0.098443   0.07532736 0.13200495 0.10014793 0.14952129 0.17723625
 0.06712006 0.06696011 0.07071596 0.0625231 ]
```

```
# Gaussian Naive Bayes
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
```

Figure 2: Output for Random Forest Classifier


LP2-A5.ipynb
File Edit View Insert Runtime Tools Help Last edited on December 3
Comment
Share
+ Code + Text
RAM
Disk
Editing

```
# Gaussian Naive Bayes
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score

gaussian = GaussianNB()
gaussian.fit(x_train, y_train)
y_pred = gaussian.predict(x_val)
acc_gaussian = round(accuracy_score(y_pred, y_val) * 100, 2)
print(acc_gaussian)
```

```
66.56
```

```
print(metrics.confusion_matrix(y_val, y_pred))
```

```
[[2258  517]
 [ 988  737]]
```

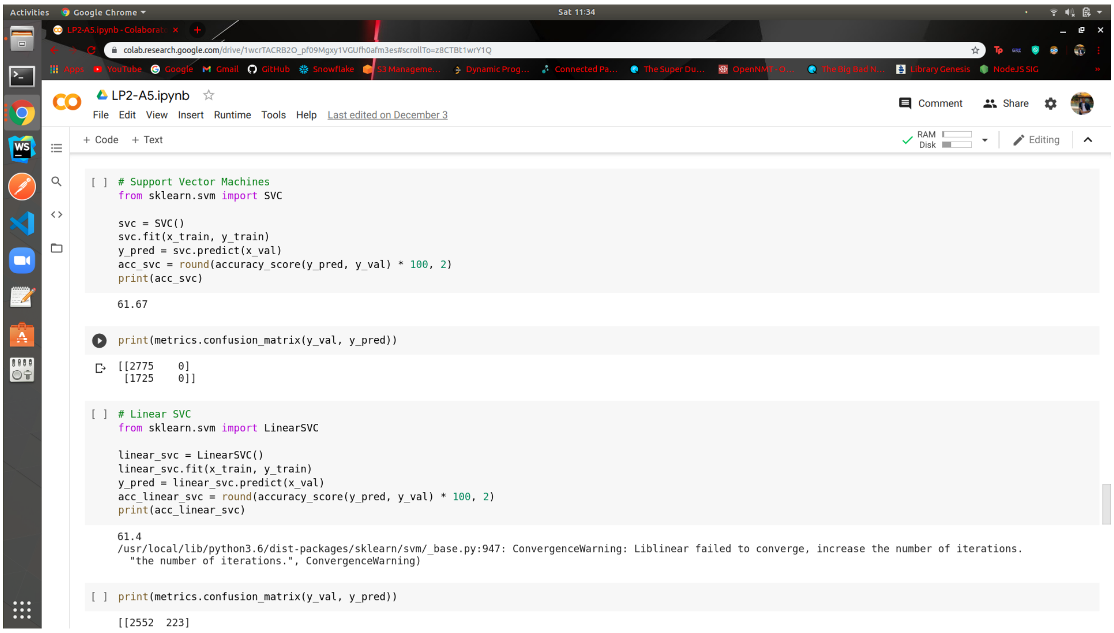
```
# Logistic Regression
from sklearn.linear_model import LogisticRegression

logreg = LogisticRegression()
logreg.fit(x_train, y_train)
y_pred = logreg.predict(x_val)
acc_logreg = round(accuracy_score(y_pred, y_val) * 100, 2)
print(acc_logreg)
```

```
65.47
```

```
print(metrics.confusion_matrix(y_val, y_pred))
```

```
[[2358  417]
```

Figure 3: Output for GaussianNB and Logistic regression classifier


LP2-A5.ipynb
File Edit View Insert Runtime Tools Help Last edited on December 3
Comment
Share
+ Code + Text
RAM
Disk
Editing

```
# Support Vector Machines
from sklearn.svm import SVC

svc = SVC()
svc.fit(x_train, y_train)
y_pred = svc.predict(x_val)
acc_svc = round(accuracy_score(y_pred, y_val) * 100, 2)
print(acc_svc)
```

```
61.67
```

```
print(metrics.confusion_matrix(y_val, y_pred))
```

```
[[2775    0]
 [1725    0]]
```

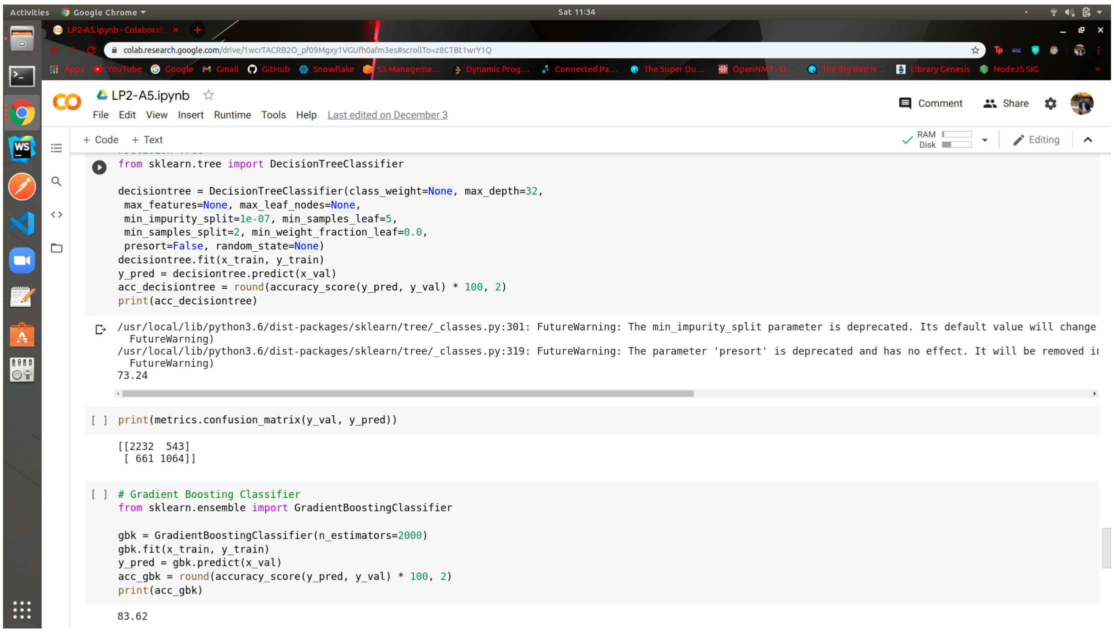
```
# Linear SVC
from sklearn.svm import LinearSVC

linear_svc = LinearSVC()
linear_svc.fit(x_train, y_train)
y_pred = linear_svc.predict(x_val)
acc_linear_svc = round(accuracy_score(y_pred, y_val) * 100, 2)
print(acc_linear_svc)
```

```
61.4
/usr/local/lib/python3.6/dist-packages/sklearn/svm/_base.py:947: ConvergenceWarning: Liblinear failed to converge, increase the number of iterations.
  "the number of iterations.", ConvergenceWarning)
```

```
print(metrics.confusion_matrix(y_val, y_pred))
```

```
[[2552  223]
```

Figure 4: Output for SVC and Linear SVC


LP2-A5.ipynb
File Edit View Insert Runtime Tools Help Last edited on December 3
Comment
Share
+ Code + Text
RAM
Disk
Editing

```
from sklearn.tree import DecisionTreeClassifier

decisiontree = DecisionTreeClassifier(class_weight=None, max_depth=32,
 max_features=None, max_leaf_nodes=None,
 min_impurity_split=1e-07, min_samples_leaf=5,
 min_samples_split=2, min_weight_fraction_leaf=0.0,
 presort=False, random_state=None)
decisiontree.fit(x_train, y_train)
y_pred = decisiontree.predict(x_val)
acc_decisiontree = round(accuracy_score(y_pred, y_val) * 100, 2)
print(acc_decisiontree)
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/tree/_classes.py:301: FutureWarning: The min_impurity_split parameter is deprecated. Its default value will change
  FutureWarning)
/usr/local/lib/python3.6/dist-packages/sklearn/tree/_classes.py:319: FutureWarning: The parameter 'presort' is deprecated and has no effect. It will be removed ir
  FutureWarning)
73.24
```

```
print(metrics.confusion_matrix(y_val, y_pred))
```

```
[[2232  543]
 [ 661 1064]]
```

```
# Gradient Boosting Classifier
from sklearn.ensemble import GradientBoostingClassifier

gbk = GradientBoostingClassifier(n_estimators=2000)
gbk.fit(x_train, y_train)
y_pred = gbk.predict(x_val)
acc_gbk = round(accuracy_score(y_pred, y_val) * 100, 2)
print(acc_gbk)
```

```
83.62
```

Figure 5: Output for Decision tree and Gradient boosting classifier

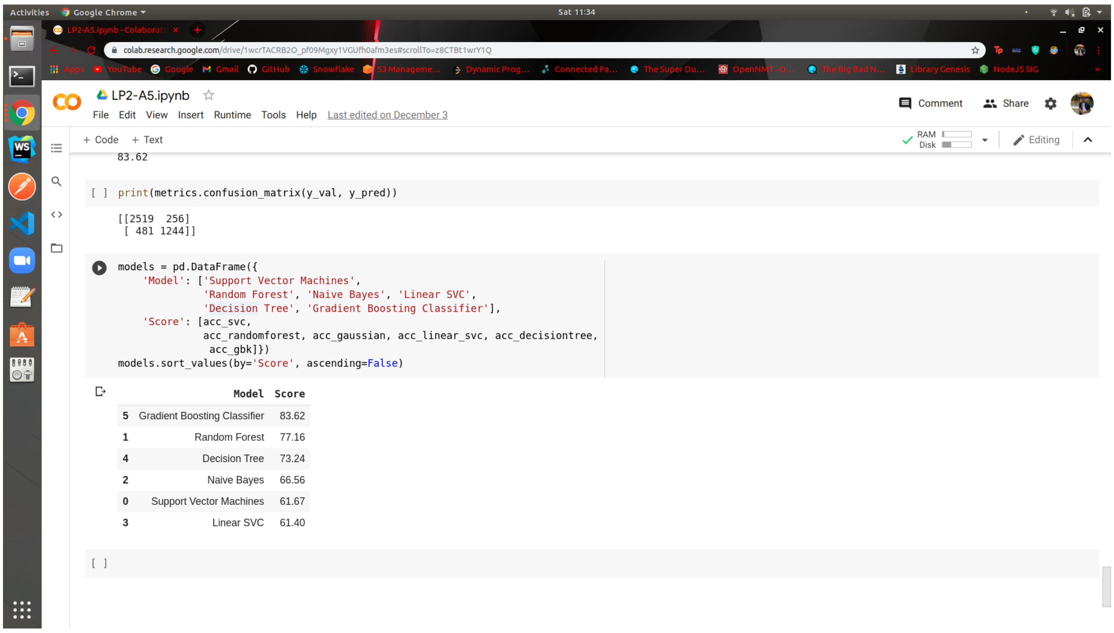
9 Result

The Accuracy for Various models are:

Model	Accuracy
DecisionTree	73.24
RandomForest	77.16
GaussianNB	66.56
LogisticRegression	65.47
SVC	61.67
LinearSVC	61.4
GradientBoostingClassifier	83.62

Table 2: Accuracy of vaious Models

We see that Gradient Boost Classifier gives the best score. We then use this model to perform training and testing of the model. After training, the model gives an accuracy of 83.62 %.


colab.research.google.com/drive/1wcrTACRB2O_pf09Mgxy1VGUfh0afm3es#scrollTo=z8CTBt1wrY1Q
LP2-A5.ipynb
File Edit View Insert Runtime Tools Help Last edited on December 3
+ Code + Text
Comment
Share
Editing

```
83.62
```

```
print(metrics.confusion_matrix(y_val, y_pred))
```

```
[[2519  256]
 [ 481 1244]]
```

```
models = pd.DataFrame({
    'Model': ['Support Vector Machines',
              'Random Forest', 'Naive Bayes', 'Linear SVC',
              'Decision Tree', 'Gradient Boosting Classifier'],
    'Score': [acc_svc,
              acc_randomforest, acc_gaussian, acc_linear_svc, acc_decisiontree,
               acc_gbk]})
models.sort_values(by='Score', ascending=False)
```

```
                        Model  Score
5  Gradient Boosting Classifier  83.62
1                 Random Forest  77.16
4                 Decision Tree  73.24
2                   Naive Bayes  66.56
0       Support Vector Machines  61.67
3                    Linear SVC  61.40
```

Figure 6: Comparison of various models

10 Conclusion

We have analysed the Soccer game dataset and performed data pre-processing steps.We have experimented multiple classification models and found out the best performer amongt them. We presented classification of soccer game results to predict the win/loss using Gradient Boost Classifier. We report a classification accuracy of 83.62

References

[1] https://scikit-learn.org/stable/modules/generated/sklearn.ensembl

[2] https://scikit-learn.org/stable/modules/generated/sklearn.ensembl

[3] https://www.kaggle.com/c/datawiz19round1/data