LOVELY PROFESSIONAL UNIVERSITY

PROJECT REPORT

INT 247

MACHINE LEARNING FOUNDATION

Submitted By:

K Nikhil Raju

nikhilkataru07@gmail.com

11707830

KM028(A23)

School of Computer Science & Engineering

Lovely Professional University, Phagwara

(March - April,2020)

GitHub link:

github.com/Nikhilkataru07/ML_ClassificationTechniques

I. Problem:

Analyze the difficulty level of course using Classification techniques

II. Software requirement analysis:

Software and Libraries

This project uses the following software and Python libraries:

Anaconda: Anaconda is a free and open-source distribution of the Python and R programming languages for scientific computing, that aims to simplify package management and deployment. We need to have this software installed to run and execute a **Jupyter Notebook.**

Jupyter Notebook is a web application that allows you to create and share documents that contain:

- live code (e.g. Python code)
- visualizations
- explanatory text (written in markdown syntax)

Jupyter Notebook is great for the following use cases:

- learn and try out Python
- data processing / transformation
- numeric simulation
- statistical modeling

Python: Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language.

NumPy: NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed.

Pandas : Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.

Scikit-learn: Scikit-learn is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

Matplotlib : Matplotlib is one of the most popular Python packages used for data visualization. It is a cross-platform library for making 2D plots from data in arrays. It provides an object-oriented API that helps in embedding plots in applications using Python GUI toolkits such as PyQt, WxPythonotTkinter. It can be used in Python and IPython shells, Jupyter notebook and web application servers also.

I have used the Anaconda distribution of Python, which already has the above packages and more included.

III. Design

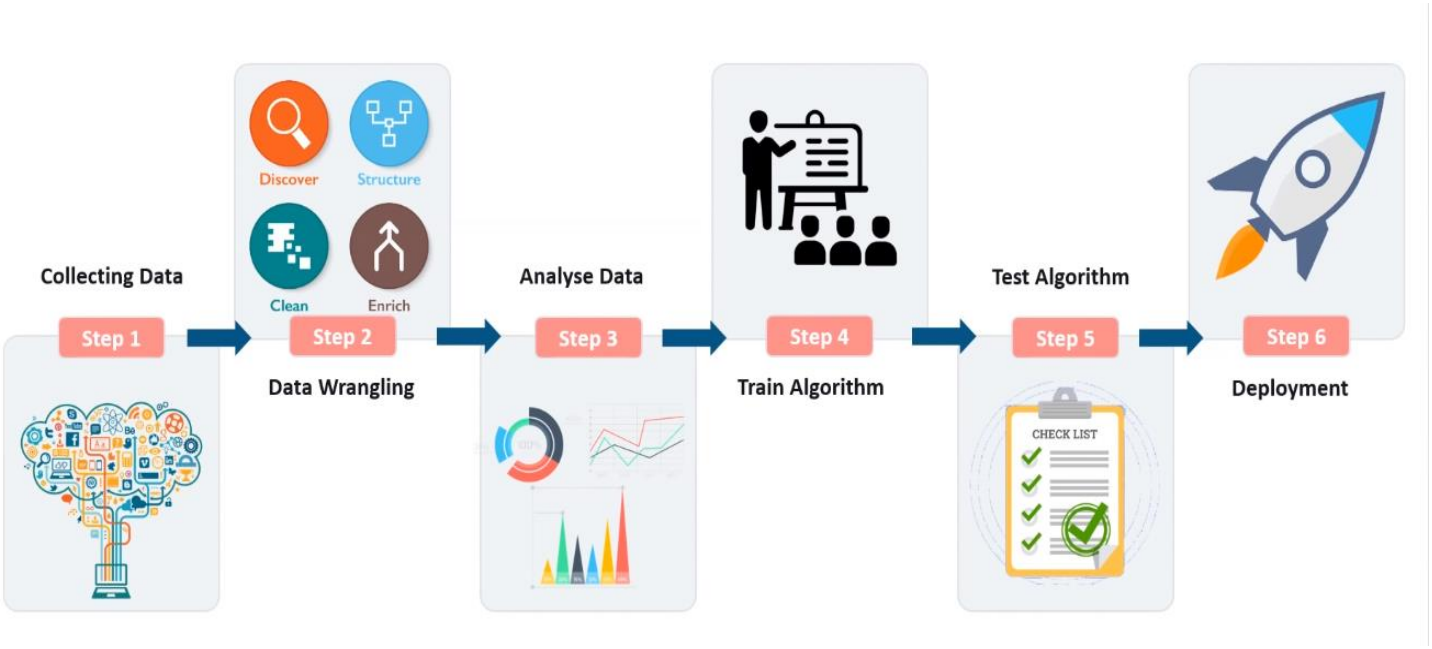


FIG 1: Life Cycle

Step1: Collecting Data – Gathering data, Bachelor of Technology (Cyber Security) Dataset

Step2: Data Wrangling – Cleaning data to have homogeneity

Step 3: Analyze Data – Data Visualization, Transforming the data into visual graphs

Step 4,5: Train & Test -Model Building & selecting the right ML algorithm

Step 6: Deployment – Gaining insights from the model's results, transforming results into visual graphs

IV.

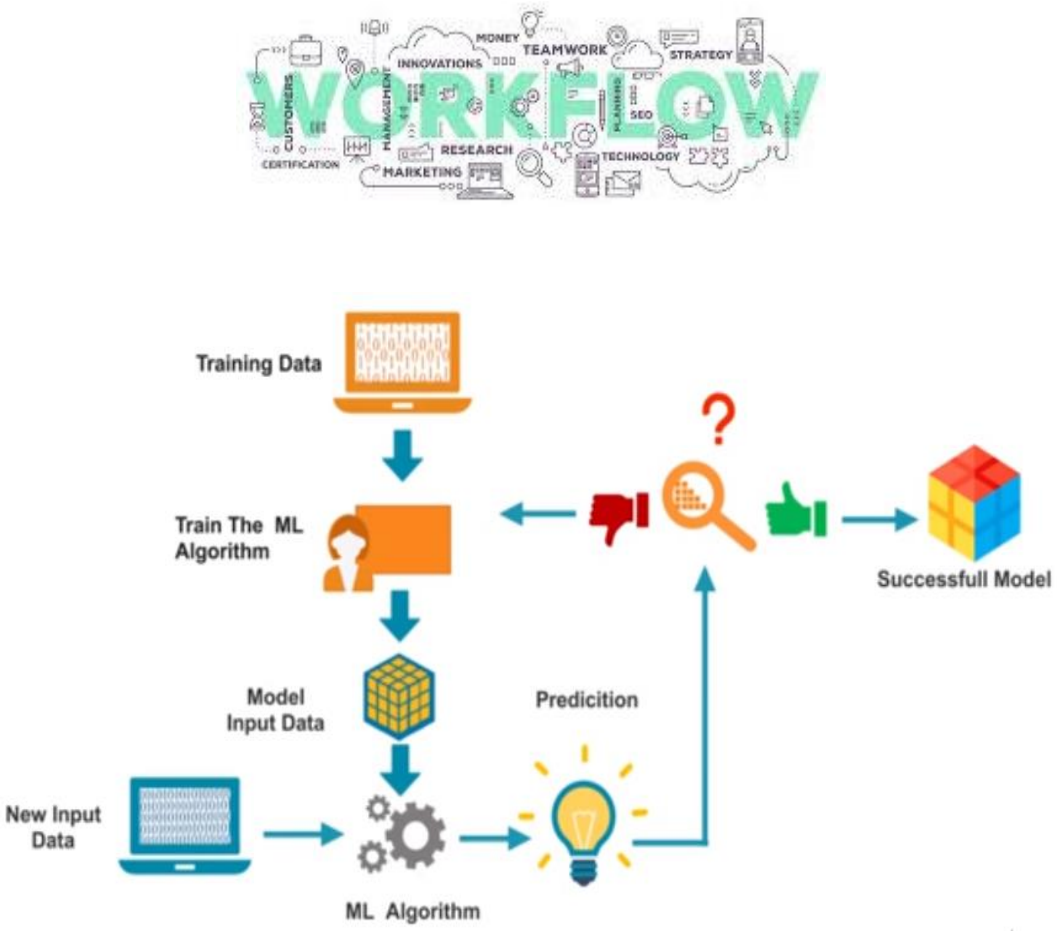


FIG 2: Workflow

V. Implementation

Data Collection

[86]:
```
#Importing Required Libraries
%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sea
```

[87]:
```
#dataset
df=pd.read_csv('dataset.csv')
df
```

t[87]:

	Termid	Regd No	Course	Grade	CA_100	MTT_50	ETT_100	ETP_100	Course_Att	MHRDName	...	CA_3	CA_4	Height	Weight	ScholarTyp
0	318192	1101776	KVY1	O	87.0	39.0	82.0	89.0	88.0	Bachelor of Science (Honours) (Agriculture)	...	1.0	0.0	181	65	Hostl
1	318192	1101776	KVY147	A+	87.0	47.0	65.0	85.0	82.0	Bachelor of Science (Honours) (Agriculture)	...	0.0	1.0	181	65	Hostl
										Bachelor of						

[88]:
```
#rows and colums of the dataset
print('Rows :'+str(df.shape[0]))
print('Columns :'+str(df.shape[1]))
print(df['MHRDName'].value_counts())
```

```
Rows :65535
Columns :22
Bachelor of Science (Honours) (Agriculture)                                  7707
Bachelor of Technology in Computer Science and Engineering (Big Data)        5493
Bachelor of Technology (Computer Science and Engineering)                    4900
Bachelor of Technology in Computer Science and Engineering (Android Application Development)    4324
Bachelor of Technology in Electronics and Communication Engineering (Internet of Things)    3917
Bachelor of Technology in Computer Science and Engineering (Data Science)    3705
Bachelor of Technology (Mechanical Engineering)                              3238
Bachelor of Technology (Electronics and Communication Engineering)           3209
Bachelor of Technology (Civil Engineering)                                   3077
Bachelor of Technology in Mechanical Engineering (Robotics and Mechatronics)    2692
Bachelor of Computer Applications                                            1719
Bachelor of Technology in Computer Science and Engineering (Machine Learning)    1716
Bachelor of Architecture                                                     1577
Bachelor of Technology in Computer Science and Engineering (Full Stack Web Developer)    1388
Bachelor of Technology in Computer Science and Engineering (Cyber Security)  1365
Diploma in Computer Science Engineering                                      1110
Bachelor of Technology (Electrical and Electronics Engineering)               981
Bachelor of Technology in Computer Science and Engineering (Cloud Computing)    882
Bachelor of Technology (Biotechnology)                                        822
Bachelor of Science (Honours) (Agriculture)                                   818
Diploma in Mechanical Engineering                                             718
Bachelor of Technology (Electrical Engineering)                               622
Diploma in Civil Engineering                                                  601
Integrated Bachelor of Science - Master of Science (Chemistry)                  4
Master of Planning (Urban)                                                      4
Bachelor of Technology in Automobile Engineering                                4
Master of Science in Agriculture Horticulture (Fruit Science)                   4
Bachelor of Technology in Information Technology                                4
Master of Science Ag. (Plant Pathology)                                         3
Master of Technology (Automobile Engineering)                                   3
Master of Science in Agriculture (Soil Science and Agriculture Chemistry)       3
Bachelor of Science (Honours) (Chemistry)                                       3
Dual Degree Bachelor of Technology (Civil Engineering) - Master of Business Administration    3
Integrated Bachelor of Science - Master of Science (Botany)                     2
Master of Science (Honours) (Biochemistry)                                      2
Master of Technology (VLSI Design)                                              2
Master of Science in Agriculture (Genetics and Plant Breeding)                  2
Master of Technology (Power Systems)                                            2
Master of Technology (Civil Engineering)                                        1
Name: MHRDName, Length: 135, dtype: int64
```

39]:
```
#filtering the required data B.Tech (Cyber Security)
df=df[df['MHRDName']=="Bachelor of Technology in Computer Science and Engineering (Cyber Security)"]
```

30]:
```
#shape of the newly formed dataset
print("Rows :"+str(df.shape[0]))
print("Columns :"+str(df.shape[1]))
```

```
Rows :1365
Columns :22
```

Data Preprocessing

91]:
```
#displaying the features and its values
df.head(10)
```

91]:

	Termid	Regd No	Course	Grade	CA_100	MTT_50	ETT_100	ETP_100	Course_Att	MHRDName	...	CA_3	CA_4	Height	Weight	ScholarType	Directio
47	218192	1113776	KHZ28	B+	57.0	17.0	56.0	NaN	93.0	Bachelor of Technology in Computer Science and...	...	12.0	1.0	167	90	Day Scholar	Ea
48	218192	1113776	KHZ51	C	21.0	25.0	59.0	NaN	83.0	Bachelor of Technology in Computer Science and...	...	0.0	1.0	167	90	Day Scholar	Ea
49	218192	1113776	KHZ52	A	96.0	NaN	NaN	56.0	89.0	Bachelor of Technology in Computer Science and...	...	5.0	58.0	167	90	Day Scholar	Ea
50	218192	1113776	KHZ53	B+	54.0	30.0	69.0	NaN	67.0	Bachelor of Technology in Computer Science and...	...	4.0	8.0	167	90	Day Scholar	Ea
										Bachelor of							

95]:
```
#Mapping the feature 'grade' with numeric values so as to calculate the difficulty level of the course
grade_map={'O':10,'A+':9,'A':8,'B+':7,'B':6,'C':5,'D':4,'E':3,'F':2}
df['Grade']=df['Grade'].map(grade_map)
df['Grade']
```

```
C:\Users\DELL\Anaconda3\lib\site-packages\ipykernel_launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
  This is separate from the ipykernel package so we can avoid doing imports until
```

95]:
```
47     7
48     5
49     8
50     7
51     4
52     8
53     5
54     6
55     5
604    9
605    8
606    8
607    8
608    7
609    8
610    8
```

[96]:
```
#Label Encoder
#coverting the string type values
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
#converting the Course & Gender features values into numeric type
df['Course']=le.fit_transform(df['Course'])
df['Gender']=le.fit_transform(df['Gender'])
#display data
df.head(5)
```

[92]:
```
#dropping the features that are not required as per the problem statement
df.drop(['Termid','Regd No','MHRDName', 'CA_1', 'CA_2','CA_3','CA_4','Height','Weight','ScholarType','Direction','CourseType',
```

```
C:\Users\DELL\Anaconda3\lib\site-packages\pandas\core\frame.py:3940: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
  errors=errors)
```

[93]:
```
#dataset after dropping features
df.head(5)
```

:[93]:

	Course	Grade	CA_100	MTT_50	ETT_100	ETP_100	Course_Att	Gender
47	KHZ28	B+	57.0	17.0	56.0	NaN	93.0	Male
48	KHZ51	C	21.0	25.0	59.0	NaN	83.0	Male
49	KHZ52	A	96.0	NaN	NaN	56.0	89.0	Male
50	KHZ53	B+	54.0	30.0	69.0	NaN	67.0	Male
51	KHZ54	D	64.0	8.0	42.0	NaN	68.0	Male

[94]:
```
#detailed Data
df.describe()
```

:[94]:

```
print(df.isnull().sum())
```

```
Course            0
Grade             0
CA_100            0
MTT_50          344
ETT_100         344
ETP_100        1021
Course_Att        0
Gender            0
dtype: int64
```

[98]:
```
#Eliminating the null values
from sklearn.preprocessing import Imputer
#strategy used here is mean which is the average of all the values belonging to that particular feature
#we can use other strategies 'most_frequent' , 'median' as well
#im=Imputer(missing_values=np.nan,strategy='most-frequent',axis=0)
#im=Imputer(missing_values=np.nan,strategy='median',axis=0)

im=Imputer(missing_values=np.nan,strategy='mean',axis=0)
df['MTT_50']=im.fit_transform(df[['MTT_50']].values)
df['ETT_100']=im.fit_transform(df[['ETT_100']].values)
df['ETP_100']=im.fit_transform(df[['ETP_100']].values)
```

64544	False	False	False	False	False	False	False	False
64545	False	False	False	False	False	False	False	False
64546	False	False	False	False	False	False	False	False

1365 rows × 8 columns

[100]:
```
#data after inserting new values
print(df.isnull().sum())
```

```
Course        0
Grade         0
CA_100        0
MTT_50        0
ETT_100       0
ETP_100       0
Course_Att    0
Gender        0
dtype: int64
```

[101]:
```
#display data
df.head(5)
```

:[101]:

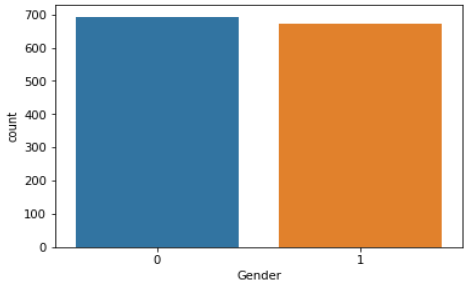
	Course	Grade	CA_100	MTT_50	ETT_100	ETP_100	Course_Att	Gender
47	3	7	57.0	17.000000	56.000000	69.270349	93.0	1

Data Visualization

Analysing Data

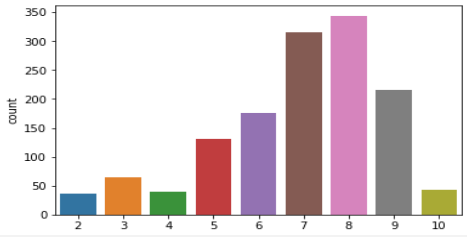
```
02]: #number of male and female
     sea.countplot(x="Gender", data=df)
     df['Gender'].value_counts()
```

```
02]: 0    693
     1    672
     Name: Gender, dtype: int64
```



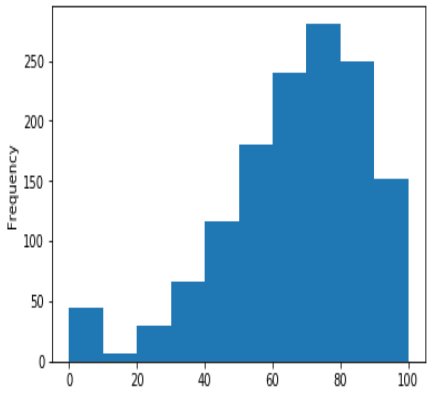

```
103]: #grades comparison
      sea.countplot(x="Grade", data=df)
      df['Grade'].value_counts()
```

```
103]: 8     344
      7     315
      9     215
      6     176
      5     131
      3      64
      10     43
      4      40
      2      37
      Name: Grade, dtype: int64
```



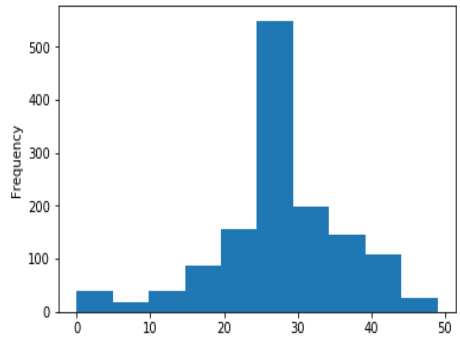

```
[108]: df["CA_100"].plot.hist()
```

```
[108]: <matplotlib.axes._subplots.AxesSubplot at 0x2673e90a4e0>
```



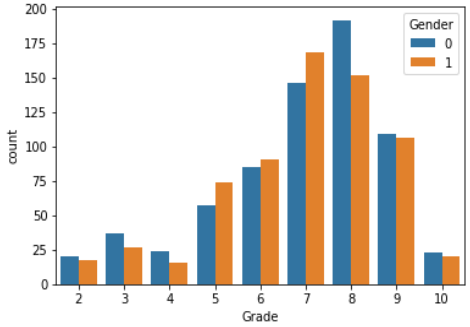
05]:
```
df["MTT_50"].plot.hist()
```

05]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2673e7eb9e8>
```



4]:
```
#Grades and gender comparison
sea.countplot(x="Grade",hue='Gender',data=df)
```

4]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2673e753f60>
```



Random Forest Classifier

121]:
```
from sklearn.ensemble import RandomForestClassifier
```

122]:
```
from sklearn.ensemble import RandomForestClassifier
ran=RandomForestClassifier(n_estimators=100,criterion='gini',max_features=5)
ran.fit(X_train,y_train)
ran_pred=ran.predict(X_test)
# comparing actual response values (y_test) with predicted response values (y_pred)
print("RandomForestClassifier")
print("Accuracy Score :")
print(accuracy_score(y_test,ran_pred))
print("r2_score :")
print(r2_score(y_test,ran_pred))
```

```
RandomForestClassifier
Accuracy Score :
0.7585365853658537
r2_score :
0.9260916590672916
```

126]:
```
print("Comparing the  accuracy values attained from the  above models, SVC is best model for the given dataset with an accuracy_
```

```
Comparing the  accuracy values attained from the  above models, SVC is best model for the given dataset with an accuracy_score
of about 0.83
```

Decision Tree Classifier

119]:
```
from sklearn.tree import DecisionTreeClassifier
```

120]:
```
deci=DecisionTreeClassifier(criterion='gini')
deci.fit(X_train,y_train)
deci_pred=deci.predict(X_test)
print("DecisionTreeClassifier")
print("Accuracy Score :")
print(accuracy_score(y_test,deci_pred))
print("r2_score :")
print(r2_score(y_test,deci_pred))
# comparing actual response values (y_test) with predicted response values (y_pred)
```

```
DecisionTreeClassifier
Accuracy Score :
0.6439024390243903
r2_score :
0.8855145307120791
```

Naive Bayes Classifier

117]:
```
from sklearn.naive_bayes import GaussianNB
```

118]:
```
gnb=GaussianNB()
gnb.fit(X_train,y_train)
gnb_pred=gnb.predict(X_test)
print("GaussianNB")
print("Accuracy Score :")
# comparing actual response values (y_test) with predicted response values (y_pred)
print(accuracy_score(y_test,gnb_pred))
print("r2_score :")
print(r2_score(y_test,gnb_pred))
```

```
GaussianNB
Accuracy Score :
0.4317073170731707
r2_score :
0.4551071335157183
```

SVC

115]:
```
from sklearn.svm import SVC
```

116]:
```
model=SVC(C=10,kernel='rbf',gamma='auto')
model.fit(X_train,y_train)
svm_pred=model.predict(X_test)
from sklearn.metrics import accuracy_score
print("SVC")
print('Accuracy Score :')
# comparing actual response values (y_test) with predicted response values (y_pred)
print(accuracy_score(y_test,svm_pred))
print("r2_score :")
print(r2_score(y_test,svm_pred))
```

```
SVC
Accuracy Score :
0.8268292682926829
r2_score :
0.9384097158894097
```

KNN Classifier

13]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import r2_score
```

14]:
```
knn=KNeighborsClassifier(n_neighbors=2)
knn.fit(X_train,y_train)
knn_pred=knn.predict(X_test)
print("KNeighborsClassifier")
print("Accuracy score :")
# comparing actual response values (y_test) with predicted response values (y_pred)
print(accuracy_score(y_test,knn_pred))
print("r2_score :")
print(r2_score(y_test,knn_pred))
```

```
KNeighborsClassifier
Accuracy score :
0.7097560975609756
r2_score :
0.8985571791119689
```

Difficulty level of the course

```
[24]: #calculating statistics
print("Calculating statistics of dataset provided in order to define the DIFFICULTY LEVEL of this course ..")
maximum_grade = np.max(target)
minimum_grade = np.min(target)
mean_grade= np.mean(target)
print("Maximum GRADE: "+str(maximum_grade))
print("Minimum GRADE: "+str(minimum_grade))
print("Mean GRADE: "+str(mean_grade))
diff=(mean_grade)*10
```

```
Calculating statistics of dataset provided in order to define the DIFFICULTY LEVEL of this course ..
Maximum GRADE: 10
Minimum GRADE: 2
Mean GRADE: 6.92967032967033
```

```
[25]: #Setting difficulty levels
print("DIFFICULTY LEVEL of this course (B.Tech -Cyber Security) whose accuracy value is {} is :".format(diff))
level=['EASY','MEDIUM','HARD']
if diff<50:
    diff=level[0]
elif diff>50 and diff<75:
    diff=level[1]
else :
    diff=level[2]

print(diff)
```

```
DIFFICULTY LEVEL of this course (B.Tech -Cyber Security) whose accuracy value is 69.2967032967033 is :
MEDIUM
```

Difficulty level of this course is MEDIUM (50% -75%)

SVC best suits the dataset with an accuracy score of about 0.83 when compared to other Classification Techniques

VI. Technical and Managerial lessons learnt:

I have learnt to

- Use NumPy to investigate the latent features of a dataset.
- Use scikit-learn, matplotlib and sklearn libraries.
- Determine the best-guess model for predictions from unseen data.
- Evaluate a model's performance on unseen data using previous data.

VII. References

www.edureka.com

www.towardsdatascience.com

www.edx.org

www.geeksforgeeks.com

www.scikit-learn.org