Project Report

Classify whether the student will pass a course or not

Course Title: Machine Learning Foundation

Course Code: INT247

Submitted By:

Name: K. Shiva Chandra

Reg No: 11709191

Section: KM028

Roll. No: 64

Under the guidance of:
Usha Mittal

ACKNOWLEDGMENT

Place: - Lovely Professional University

Date: - 8th April, 2020

I would like to thank Usha Mittal for assigning us with this project. Through the project, I can grasp more technical and have a hands-on practical experience with python and some machine learning algorithms. Through it, I can learn how a project is created and how necessary and crucial technical knowledge is. I am grateful to the faculty that has provided me with the necessary guidelines.

Registration No: 11709191.

Name: K. Shiva Chandra.

DECLARATION STATEMENT

Place: - Lovely Professional University

Date: - 8th April, 2020

I hereby declare that this report has been written by me. No part of the report is copied from the other sources. All information included from other sources has been duly acknowledged. I aver that if any part of the report is found to be copied, I will take full responsibility for it.

Name: K. Shiva Chandra.

Reg No: 11709191.

TABLE OF CONTENTS

INTRODUCTION

- This is the project I have been assigned to in INT247.
- Name: Classify whether the student will pass a course or not.
- Dataset: It consists may course but we have predicted only on course Dual Degree Bachelor of Technology - Master of Technology (Mechanical Engineering)

Collection of Database/necessary requirements for project:

- Database of all the students necessary for the project is given in an csv file with every other detail.
- It will be attached with this report in the zip folder.
- Anaconda Application must be Installed in your System.

IMPLEMENTATION OF CODE

- Data Pre-processing
- Choosing appropriate model
- Training(Retraining for optimization)
- Evaluating Accuracy with confusion matrix
- Hyper parameters tuning
- Prediction on new data

1. Data Pre-Processing

- The total dataset contains 65535 rows and 22 columns, you can see the different columns in Fig 1.
- You can see in fig 2 that there are 135 unique courses but we want to perform classification on only one course and that is Dual Degree Bachelor of Technology - Master of Technology (Mechanical Engineering).
- There are only 8 rows on the course we have classify, so we can't perform any classification on such a small dataset, so I am training the model on all other courses and making predictions the course we have to classify.
- From fig 3 you can see that there are 16 different types of grades, but our work is to predict whether the student will pass a course or not, so we going to add a new column based on the grade column with help of the code in fig 4. We are just appending to fail when the grades are ('ReApp', 'Fail', 'F', 'E') and passed otherwise.
- And out of 22 columns we are having, most of the columns are not required for us so we are reducing the number of features in one way called feature selection in another way called dimensionality reduction. As in the fig 5. we are dropping ['Height', 'Weight', 'Direction', 'Course', 'Termid', 'Regd No', 'Gender', 'ProgramType' , 'ScholarType', 'Medium', 'CourseType', 'MHRDName', 'Grade'] .

- And out of 65535 rows, nearly 35874 rows contain null values fig 7 and from fig 6 we can see that the data is not scaled so we try to remove both of these problems.
- For removing the null values we are taking the help of SimpleImputer() function available in the sklearn.preprocessing and defining the strategy as mean.
- And for scaling the data we are using the StandardsScalar library. And we can see that after using standard scalar we can see in fig 8 that the standard deviation for every column is 1.
- And at last before training on the model we have to split the data into training data, testing data and validation data. The validation data is the data of the course we have to predict at last. We can do that by using train_test_split as in the fig 9 and for the validation data we can do that as in the fig 10.

```
In [3]:  #data.shape gives us
         data.shape

Out[3]:  (65535, 22)
```

Fig 1.

```
In [6]:  #In this cell, I am trying to get
         data['MHRDName'].nunique()

Out[6]:  135
```

Fig 2.

```
In [9]:  #In this cell we trying to get unique grades
         data['Grade'].unique()

Out[9]:  array(['O', 'A+', 'B+', 'A', 'F', 'E', 'C', 'D', 'B', 'R', 'I', 'FAIL',
                'ReApp', 'PASS', 'M', 'S'], dtype=object)
```

Fig 3.

```
In [15]: #In this cell I am a new columns which will contain either pass or fail
         abc=[]
         for i in range(0,data.shape[0]):
             if(data.iloc[i,3])=='ReApp' or (data.iloc[i,3])=='FAIL' or (data.iloc[i,3])=='F' or (data.iloc[i,3])=='E':
                 abc.append(str('Fail'))
             else:
                 abc.append(str('Passed'))
```

Fig 4.

```
In [29]: cols=['Height','Weight','Direction','Course','Termid','Regd No','Gender','ProgramType'
               ,'ScholarType','Medium','CourseType','MHRDName','Grade']
         df=df.drop(cols,axis=1)
```

Fig 5.

```
In [13]: #data.describe() the function shows us the different values of the statistical functions applied to the data frame.
         data.describe()

Out[13]:
```

	Termid	Regd No	CA_100	MTT_50	ETT_100	ETP_100	Course_Att	CA_1	CA_2	CA_3	
count	65535.000000	6.553500e+04	62969.000000	38414.000000	39699.000000	29644.000000	59454.000000	62969.000000	62969.000000	62969.000000	62969.0
mean	288099.682918	8.450856e+06	63.772317	26.110637	52.052470	67.181892	81.046692	31.961918	15.926504	7.937985	7.9
std	84391.200813	4.155810e+06	23.809873	11.811316	22.972317	22.770638	17.960987	23.197636	16.421255	10.651955	10.6
min	118192.000000	1.101776e+06	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0

Fig 6

```
In [11]: #In this cell we are got
         data.isnull().sum()

Out[11]: Termid             0
         Regd No            0
         Course             0
         Grade              0
         CA_100          2566
         MTT_50         27121
         ETT_100        25836
         ETP_100        35891
         Course_Att      6081
         MHRDName           0
         CA_1            2566
         CA_2            2566
         CA_3            2566
         CA_4            2566
         Height             0
         Weight             0
         ScholarType        0
         Direction          0
         Gender             0
         Medium             0
         CourseType         0
         ProgramType        0
         dtype: int64
```

Fig 7

```
In [54]: result.describe()
Out[54]:
```

	CA_100	MTT_50	ETT_100	ETP_100	Course_Att	CA_1	CA_2	CA_3	CA_4
count	6.553500e+04	6.553500e+04	6.553500e+04	6.553500e+04	6.553500e+04	6.553500e+04	6.553500e+04	6.553500e+04	6.553500e+04
mean	1.745519e-15	-1.782266e-15	-2.145025e-16	6.155271e-16	-3.926856e-15	9.516526e-16	-7.370434e-16	-6.777959e-16	-1.110574e-15
std	1.000008e+00	1.000008e+00	1.000008e+00	1.000008e+00	1.000008e+00	1.000008e+00	1.000008e+00	1.000008e+00	1.000008e+00

Fig 8

```
In [57]: from sklearn.model_selection import train_test_split
         x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=54)
```

Fig 9

```
In [25]: validation=df[df['MHRDName']=='Dual Degree Bachelor of Technology - Master of Technology (Mechanical Engineering)']
```

Fig 10

2. Choosing appropriate model

- As we have to perform classification, I am choosing three famous models like Logistic Regression, Decision Tree, VotingClassifier and ANN.
- The implementation of Logistic Regression can be seen in fig 11. The solver=liblinear follows the l1 penalty and the c value represents Inverse of regularization strength. The lower the c value the stronger the regularization.
- Now another classifier Decision Tree (fig 12) we set all parameters as default except the criteria we are filling the criteria as Gini. I experimented with entropy as well but there is not much difference in the accuracy.
- The other model is the voting classifier i.e. the combination of SVM and Decision tree and logistic regression (fig 13).

```
In [62]: from sklearn.linear_model import LogisticRegression
         lr3=LogisticRegression(solver='liblinear',C=0.02)

In [63]: lr3.fit(x_train,y_train)
         y_pred3=lr3.predict(x_test)

In [64]: from sklearn.metrics import accuracy_score
         print(accuracy_score(y_test,y_pred3))

         0.9742885481040665
```

Fig 11

```
In [61]: from sklearn.tree import DecisionTreeClassifier
         dt=DecisionTreeClassifier()

In [62]: dt.fit(x_train,y_train)
         y_pred=dt.predict(x_test)

In [63]: from sklearn.metrics import accuracy_score
         accuracy_score(y_pred,y_test)

Out[63]: 0.9788662546730754

In [64]: from sklearn.metrics import confusion_matrix
         confusion_matrix(y_pred,y_test)

Out[64]: array([[ 1398,   161],
                [  116, 11432]], dtype=int64)
```

FIG 12

```
In [139]: from sklearn.svm import SVC
          from sklearn.tree import DecisionTreeClassifier
          from sklearn.linear_model import LogisticRegression
          from sklearn.ensemble import VotingClassifier
          svm=SVC(kernel='linear',C=1.0)
          lr1=LogisticRegression(solver='liblinear',C=0.02)
          dt1=DecisionTreeClassifier()
          vc=VotingClassifier(estimators=[('lr',lr1),('dt',dt1),('svm',svm)])

In [140]: vc.fit(x_train,y_train)

Out[140]: VotingClassifier(estimators=[('lr',
                                        LogisticRegression(C=0.02, class_weight=None,
                                                           dual=False, fit_intercept=True,
                                                           intercept_scaling=1,
                                                           l1_ratio=None, max_iter=100,
```

FIG 13

3. Training (Retraining for optimization):

- Instead of training the model on epochs, I used cross-validation to train the model with 10 as cv parameter i.e. our model is trained on the data ten times by increasing the training size gradually.
- You can see how I implemented cross-validation in fig 14 15 and 16 respectively for three different models.

```
In [65]: from sklearn.linear_model import LogisticRegression
         lr=LogisticRegression()

In [66]: from sklearn.model_selection import learning_curve
         from sklearn.linear_model import LogisticRegression
         train_sizes, train_scores, validation_scores = learning_curve(estimator =lr,
                                                             X = X,
                                                             y = y,
                                                             train_sizes = train_sizes,
                                                             cv = 8,
                                                             random_state=95,
                                                            )
```

Fig 14

```
In [127]: from sklearn.tree import DecisionTreeClassifier
          dt=DecisionTreeClassifier(criterion='gini')

In [128]: dt

Out[128]: DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
                      max_features=None, max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, presort=False,
                      random_state=None, splitter='best')

In [75]: train_sizes =np.linspace(0.1,1,10)
         X=x_train
         y=y_train
         train_sizes, train_scores, valid_scores = learning_curve( dt, X, y, train_sizes=train_sizes, cv=10)
```

Fig 15

```
In [143]: from sklearn.model_selection import validation_curve
          train_sizes =np.linspace(0.1,1,10)
          X=x_train
          y=y_train
          train_sizes, train_scores, valid_scores = learning_curve( vc, X, y, train_sizes=train_sizes, cv=10,scoring='f1')
```

Fig 16

4. Evaluating Accuracy with confusion matrix:

- In classification problems, we use the confusion matrix for evaluating how our model is predicting. In binary classification, we have 4 boxes. They are True Negative(TN), True Positive(TP), False Negative(FN), False Positive(FP).

	Predicted 0	Predicted 1
Actual 0	TN	FP
Actual 1	FN	TP

- Now we will the confusion matrix performed on our Testing set (fig 17) and on the validation set (fig 18) i.e. the course we have to predict. From the fig 18 we can see that our model gave 100% accuracy.

```
In [146]: from sklearn.metrics import confusion_matrix
          cm=confusion_matrix(y_test,y_pred)
          cm

Out[146]: array([[ 1330,   201],
                 [  108, 11468]], dtype=int64)

In [73]: from sklearn.metrics import classification_report
         print(classification_report(y_test,y_pred))

              precision    recall  f1-score   support

         0.0       0.92      0.86      0.89      1531
         1.0       0.98      0.99      0.99     11576

    accuracy                           0.97     13107
   macro avg       0.95      0.92      0.94     13107
weighted avg       0.97      0.97      0.97     13107
```

Fig 17

```
In [158]: cm_for_validation=confusion_matrix(validation_y,y_pred)
          cm_for_validation

Out[158]: array([[3, 0],
                 [0, 5]], dtype=int64)

In [160]: print(classification_report(validation_y,y_pred))

              precision    recall  f1-score   support

           0       1.00      1.00      1.00         3
           1       1.00      1.00      1.00         5

    accuracy                           1.00         8
   macro avg       1.00      1.00      1.00         8
weighted avg       1.00      1.00      1.00         8
```

Fig 18

5. Hyper parameters tuning:

➢ From figure 14 15 and 16, we have seen that we used the learning curve to check how our model worked, so now in figure 19 20 21 22, we will how our models predicted through graphs.

```
import matplotlib.pyplot as plt

plt.style.use('seaborn')
plt.plot(train_sizes, train_scores_mean, label = 'Training error')
plt.plot(train_sizes, validation_scores_mean, label = 'Validation error')
plt.legend()
plt.show()
```

Fig 19

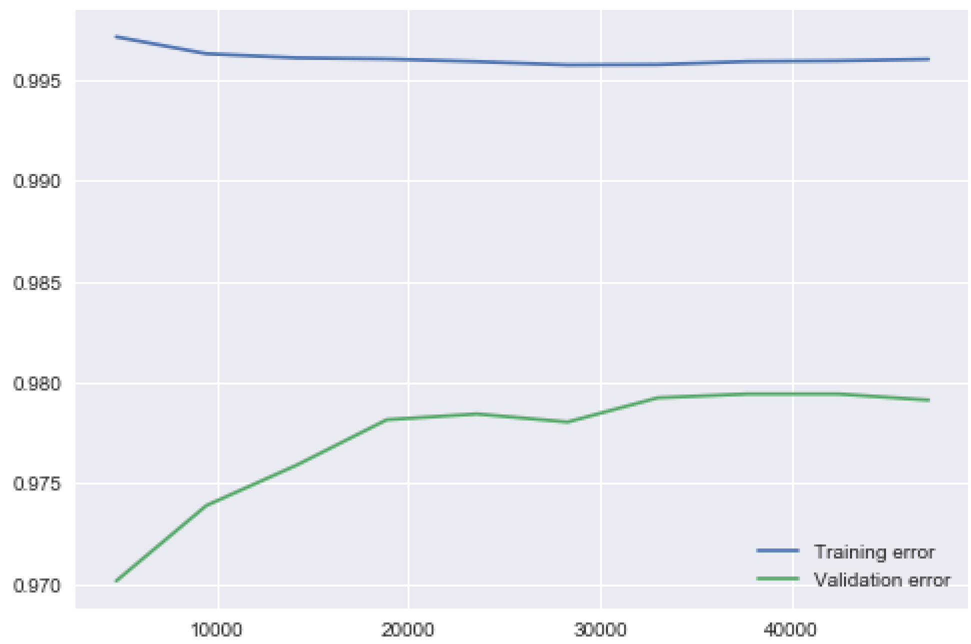

Fig 20

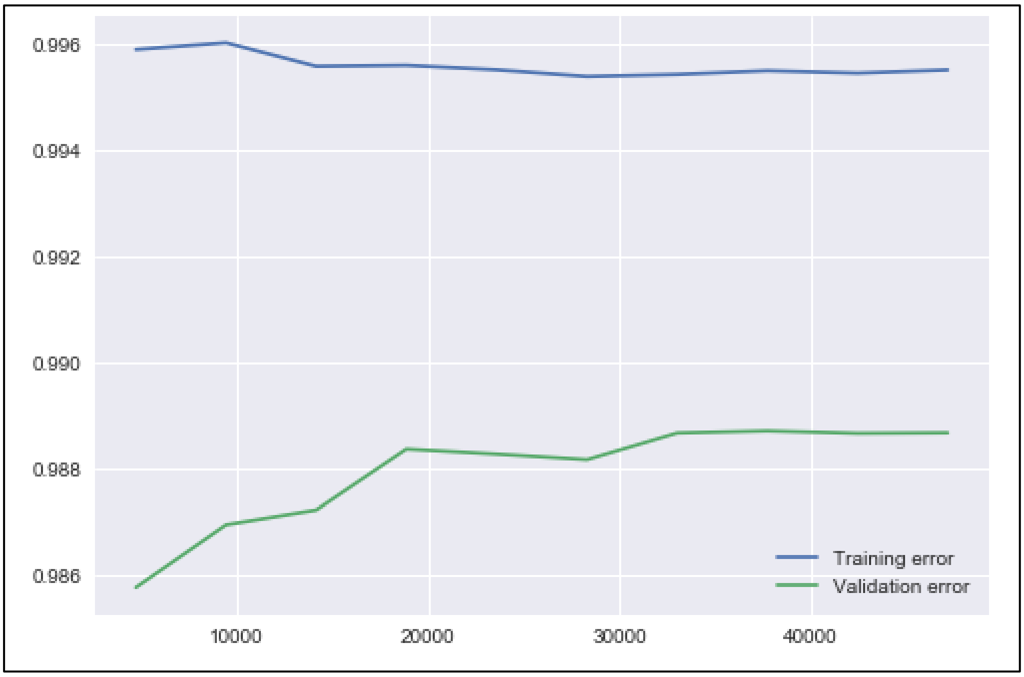
0.996
0.994
0.992
0.990
0.988
0.986
10000
20000
30000
40000
Training error
Validation error

Fig 21

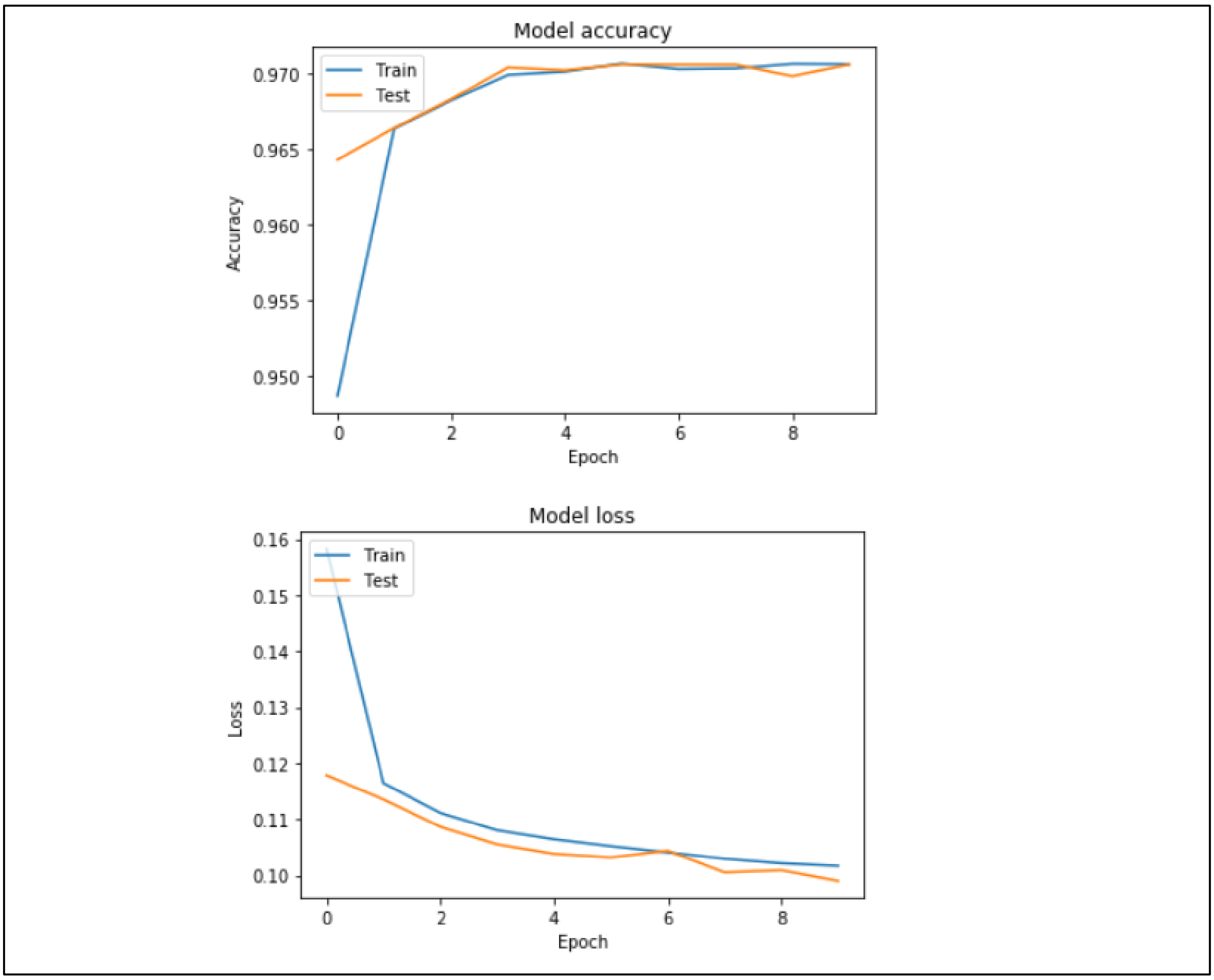
Model accuracy
Train
Test
Accuracy
0.970
0.965
0.960
0.955
0.950
0
2
4
6
8
Epoch
Model loss
Train
Test
Loss
0.16
0.15
0.14
0.13
0.12
0.11
0.10
0
2
4
6
8
Epoch

Fig 22

TECHNOLOGIES USED

❖ Python

- Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. It's high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components.

❖ Machine learning

- Machine learning is closely related to computational statistics, which focuses on making predictions using computers. The study of mathematical optimization delivers methods, theory and application domains to the field of machine learning.

❖ Classification

- Classification is one of the most widely used techniques in machine learning, with a broad array of applications, including sentiment analysis, ad targeting, spam detection, risk assessment, medical diagnosis and image classification. The core goal of classification is to predict a category or class Y from some inputs X.

❖ 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.

❖ Numpy

- NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

❖ Matplotlib

- Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+.

❖ SKlearn

- 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 and others.

❖ TensorFlow

- TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

- **Seaborn**
  - Seaborn is a Python data visualization library based on Matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.
- **Keras**
  - Keras is an open-source neural network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, R, Theano, or Plaid ML. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible.

INNOVATION/NEWNESS MADE IN PROJECT

- Self-created Video is attached in the zip folder.
- Explained the used python library/necessity/ purpose in ipynb file.
- Executable Code should be self-explanatory using comment lines in ipynb file.
- Explanation of each modules/ files/used in project in ipynb file.
- Link for GitHub: https://github.com/shivachandrakante/Project-3rd-Year-sem-2

REFERENCES

I went through different websites and textbooks to learn those concepts that will be used in making this project.

Some of the websites are:

- https://www.google.com/
- https://www.python.org/
- https://numpy.org/
- https://pandas.pydata.org/
- https://matplotlib.org/
- https://scikit-learn.org/
- https://www.tensorflow.org/
- https://seaborn.pydata.org/
- https://keras.io/