DA-MiniProject

Roll no-41232,41239

Problem Statement : Trip History Analysis Use trip history dataset that is from a bike sharing service in the United States. The data is provided quarter-wise from 2010 (Q4) onwards.
Each file has 7 columns. Predict the class of user.Make use of at least two classification algorithms and provide comparative analysis.

Abstract :

1. The bike sharing companies basically gather the data of different types of users and use it to track which members are taking the rental bikes and to which location.

2.By using this data they can decide the to increase thebikes for particular route,give some discounts to regular members.

3.Using this data we are performing analysis to find out member type of user.

S/W and H/W : Python,Jupyter,Libraries of Python Numpy,Pandas,matplotlib,64 bit OS ,8 GB RAM,1 TB HDD.

Introduction :

1.In this project we are going to analyze the Capital Bike Share Dataset and perform analysis on the same.

2.When a rental occurs within the system software collects basic data about the trip. That data can be exported from our system and used for various types of analysis or research.

Objective : We have to perform analysis on trip history dataset and predict the type of member whether casual or member.

System Architecture :

1. System architecture basically includes the Jupyter Notebook, Python and the dataset.The System basically calculates the accuracy for different classification algorithm and compares the same.

2.Each .csv file contains data for one quarter of the year. Within each file there are 7 columns.

Duration - Duration of trip
Start date – Includes start date and time
End date – Includes end date and time
Start station – Includes starting station name and number
End station – Includes ending station name and number
Bike # - Includes ID number of bike used for the trip
Member Type – Lists whether user was a Registered (annual or monthly) or Casual (1 to 5 day) member.

3.So using the above features and correaltion matrix we found out the features useful for predicting to member-type like station number,end station and duration.

4.Using the different algorithm we can find out the accuracy and compare it.

CLASSIFICATION ALGORITHMS :

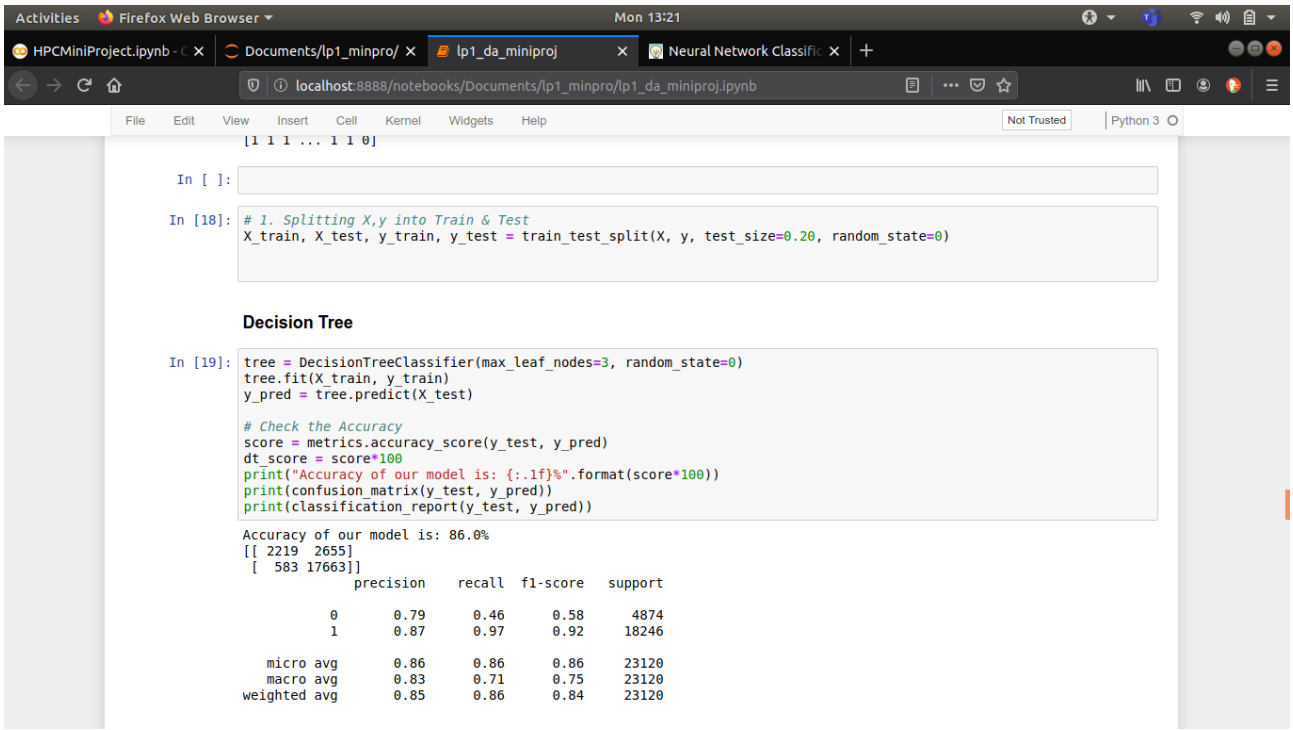
1) Decision Tree :
1. Decision tree algorithm falls under the category of supervised learning. They can be used to solve both regression and classification problems.

2. Decision tree uses the tree representation to solve the problem in which each leaf node corresponds to a class label and attributes are represented on the internal node of the tree.

3. We can represent any boolean function on discrete attributes using the decision tree.

At the beginning, we consider the whole training set as the root.

- Feature values are preferred to be categorical. If the values are continuous then they are discretized prior to building the model.
- On the basis of attribute values records are distributed recursively.
- We use statistical methods for ordering attributes as root or the internal node.



2) K-Nearest Neighbour

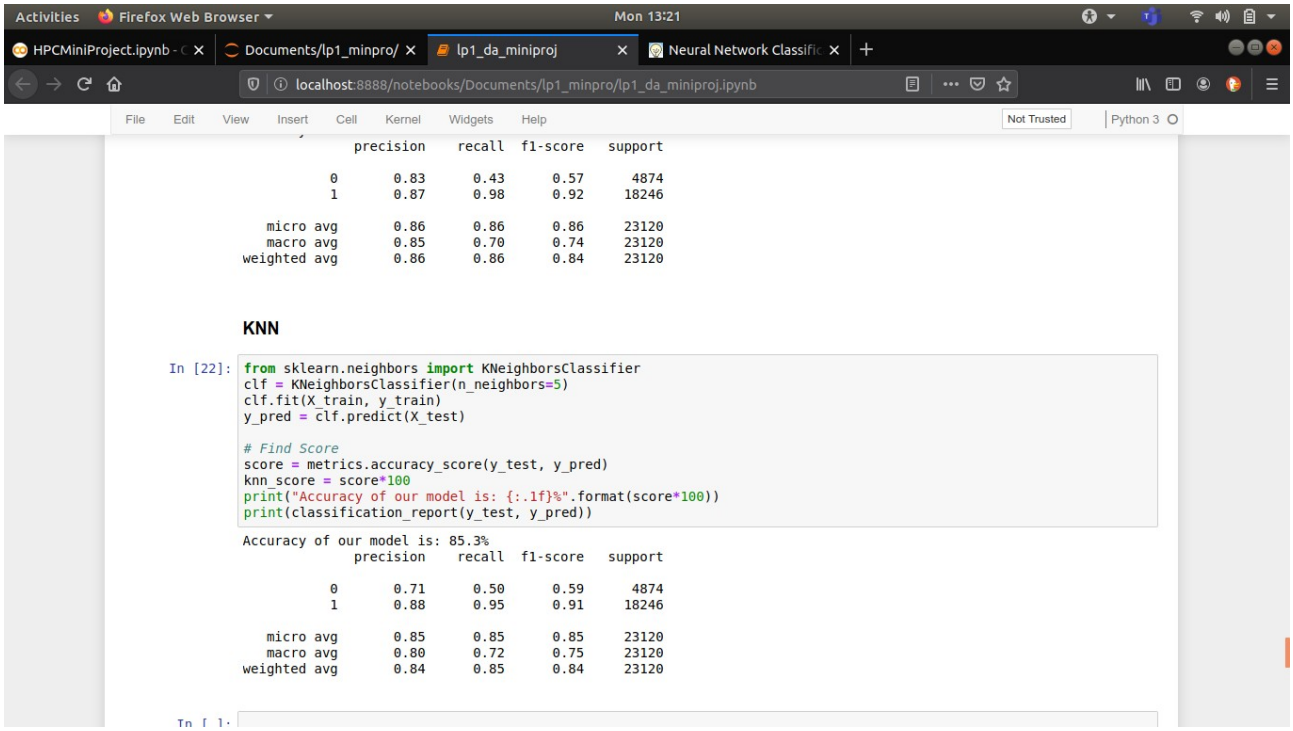
1.Neighbours based classification is a type of lazy learning as it does not attempt to construct a general internal model, but simply stores instances of the training data.

2.Classification is computed from a simple majority vote of the k nearest neighbours of each point

3.This algorithm is simple to implement, robust to noisy training data, and effective if training data is large.

4.Based on the value of k we get different accuracy so inorder to get good result with this algorithm the K value choosen should be correct.

5.KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.



3) Neural Network Classifier

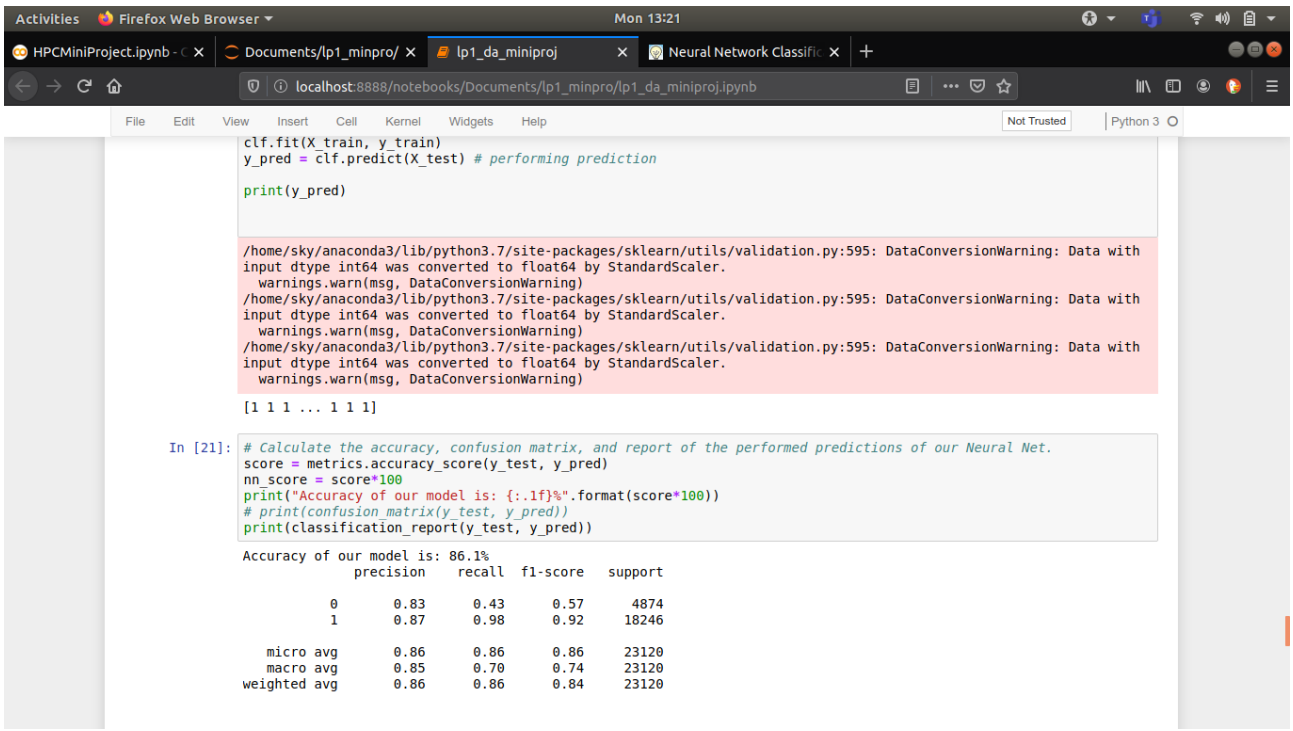
1.Neural networks are complex models, which try to mimic the way the human brain develops classification rules.

2.A neural net consists of many different layers of neurons, with each layer receiving inputs from previous layers, and passing outputs to further layers.

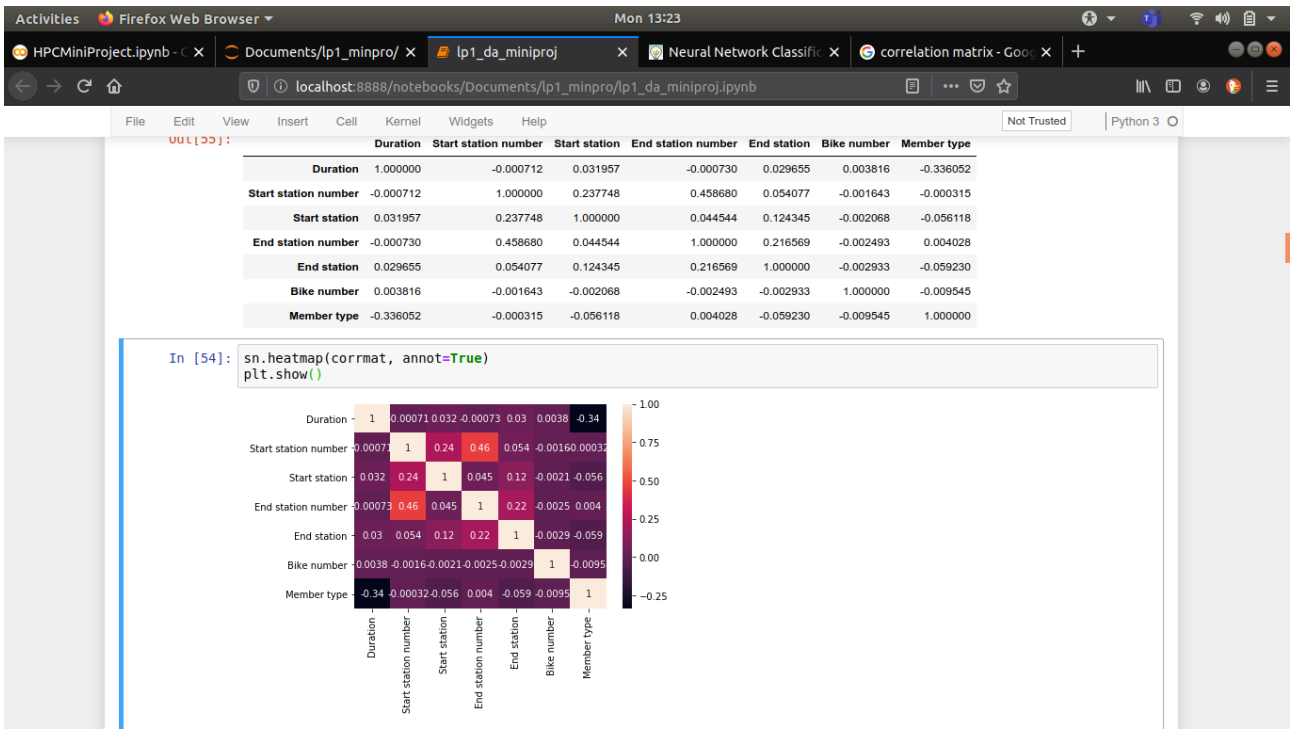
3.A neuron in an artificial neural network is
a. A set of input values (xi) and associated weights (wi).

b. A function (g) that sums the weights and maps the results to an output (y).

4.In the training phase, the correct class for each record is known (termed supervised training), and the output nodes can be assigned correct values -- 1 for the node corresponding to the correct class, and 0 for the others.

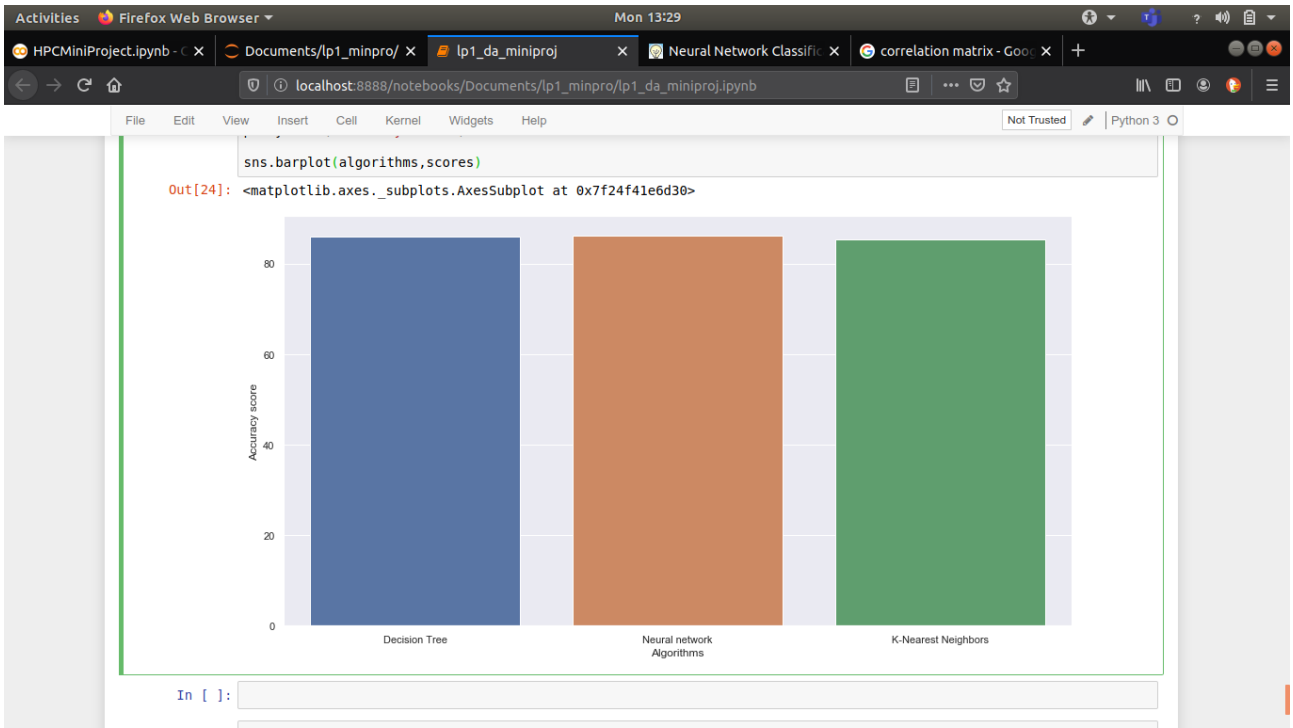


Correlation Matrix - A **correlation matrix** is a table showing **correlation** coefficients between variables. Each cell in the table shows the **correlation** between two variables. A **correlation matrix** is used to summarize data, as an input into a more advanced analysis, and as a diagnostic for advanced analyses.



Conclusion – We used different classification algorithms and the results obtained were
The accuracy score achieved using Decision Tree is: 85.99 %
The accuracy score achieved using Neural network is: 86.14 %
The accuracy score achieved using K-Nearest Neighbors is: 85.26 %

Based on the above results we can conclude that the accuracy is higher for neural network classification algorithm and can further be increased by inclreaseing the dataset size and a bit of preprocessing of data.Also we have found out that the Member type is dependent on the Start Station,End Station and the Duration.



Code & Output-

1) Import Libraries

```
import pandas as pd
import numpy as np                    # For mathematical calculations
import matplotlib.pyplot as plt       # For plotting graphs
%matplotlib inline
import seaborn as sns # For data visualization

from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import classification_report, confusion_matrix
```

2) Read Data

```
data = pd.read_csv('tripdata.csv')
```

3)Get Data Description

```
data.describe()

#Exploring the Target Variable. Our target variable is the Member Type column
data['Member type'].value_counts()

corrmat=data.corr()
```

4) Draw Correlation Matrix

```
sn.heatmap(corrmat, annot=True)
plt.show()
```

5) Convert Categorical values to Numerical

```
#Convert the Categorical Values to Numerical to allow us perform plotting
#library LabelEncoder
from sklearn.preprocessing import LabelEncoder
#Create a list with categorical predictors
cat_var =['Start station','End station','Bike number','Member type']
le = LabelEncoder()
#A for loop to transform the categorical values to numerical values
for n in cat_var:
    data[n] = le.fit_transform(data[n])
```

6) TRAINING

```
# Assign X and y
X = data.iloc[:, [0, 3, 5]].values
y = data.iloc[:, -1].values

# 1. Splitting X,y into Train & Test
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=0)
```

7) Decision Tree

```
tree = DecisionTreeClassifier(max_leaf_nodes=3, random_state=0)
tree.fit(X_train, y_train)
y_pred = tree.predict(X_test)

# Check the Accuracy
score = metrics.accuracy_score(y_test, y_pred)
dt_score = score*100
print("Accuracy of our model is: {:.1f}%".format(score*100))
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
Accuracy of our model is: 86.0%
[[ 2219  2655]
 [  583 17663]]
              precision    recall  f1-score   support

           0       0.79      0.46      0.58      4874
           1       0.87      0.97      0.92     18246

   micro avg       0.86      0.86      0.86     23120
   macro avg       0.83      0.71      0.75     23120
weighted avg       0.85      0.86      0.84     23120
```

7) Neural Network

```
from sklearn.neural_network import MLPClassifier

from sklearn.preprocessing import StandardScaler


# Scaling improvised the model's results. So, scale the data.

scaler = StandardScaler()

scaler.fit(X_train)

X_train = scaler.transform(X_train)

X_test = scaler.transform(X_test)
```

```
# Train the Neural Net.

clf = MLPClassifier(solver='lbfgs', alpha=1e-5, hidden_layer_sizes=(5, 3), random_state=1)

clf.fit(X_train, y_train)

y_pred = clf.predict(X_test) # performing prediction



print(y_pred)

# Calculate the accuracy, confusion matrix, and report of the performed predictions of our Neural
Net.

score = metrics.accuracy_score(y_test, y_pred)

nn_score = score*100

print("Accuracy of our model is: {:.1f}%".format(score*100))

# print(confusion_matrix(y_test, y_pred))

print(classification_report(y_test, y_pred))
```

Accuracy of our model is: 86.1%

	precision	recall	f1-score	support
0	0.83	0.43	0.57	4874
1	0.87	0.98	0.92	18246
micro avg	0.86	0.86	0.86	23120
macro avg	0.85	0.70	0.74	23120
weighted avg	0.86	0.86	0.84	23120

8) KNN

```
from sklearn.neighbors import KNeighborsClassifier

clf = KNeighborsClassifier(n_neighbors=5)
```

```
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)


# Find Score

score = metrics.accuracy_score(y_test, y_pred)

knn_score = score*100

print("Accuracy of our model is: {:.1f}%".format(score*100))

print(classification_report(y_test, y_pred))
```

Accuracy of our model is: 85.3%

	precision	recall	f1-score	support
0	0.71	0.50	0.59	4874
1	0.88	0.95	0.91	18246
micro avg	0.85	0.85	0.85	23120
macro avg	0.80	0.72	0.75	23120
weighted avg	0.84	0.85	0.84	23120

8) Compairing All algorithm

```
import seaborn as sns

sns.set(rc={'figure.figsize':(15,8)})

plt.xlabel("Algorithms")

plt.ylabel("Accuracy score")


sns.barplot(algorithms,scores)
```

Activities Firefox Web Browser Mon 13:29

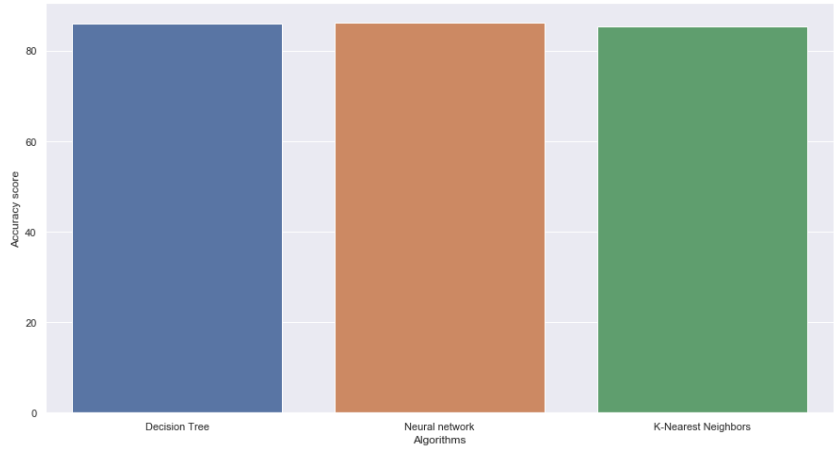
HPCMiniProject.ipynb - C | Documents/lp1_minpro/ | lp1_da_miniproj | Neural Network Classific | correlation matrix - Goo

localhost:8888/notebooks/Documents/lp1_minpro/lp1_da_miniproj.ipynb

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3

```
sns.barplot(algorithms,scores)
```

Out[24]: <matplotlib.axes._subplots.AxesSubplot at 0x7f24f41e6d30>



In [ ]: