**Assignment - 3**

1. (Titanic Dataset)

1. Find the correlation between ‘survived’ (target column) and ‘sex’ column for the Titanic use case in class.

a. Do you think we should keep this feature?

2. Do at least two visualizations to describe or show correlations.

3. Implement Naïve Bayes method using scikit-learn library and report the accuracy.

import pandas as pd

import seaborn as sns

from sklearn import preprocessing

import matplotlib.pyplot as plt

import warnings

warnings.filterwarnings("ignore")

data=pd.read\_csv("train.csv")

data.head()

le = preprocessing.LabelEncoder()

data['Sex'] = le.fit\_transform(data.Sex.values)

data['Survived'].corr(data['Sex'])

**Yes, we should keep the sex and survived feature which helps while classifying the data. So, without the sex and surivevd it would vary accuracy.**

matrix = data.corr()

print(matrix)

data.corr().style.background\_gradient(cmap="Blues")

sns.barplot(x='Sex', y='Survived', data=data)

axes = sns.factorplot('Sex','Survived',

data=data, aspect = 2.5, )

FacetGrid = sns.FacetGrid(data, row='Survived', size=4.5, aspect=1.6)

FacetGrid.map(sns.pointplot,'Survived', 'Sex', palette=None, order=None, hue\_order=None )

FacetGrid.add\_legend()

sns.heatmap(matrix, annot=True, vmax=1, vmin=-1, center=0, cmap='vlag')

plt.show()

#NaÃƒÂ¯ve Bayes method

train\_raw = pd.read\_csv('train.csv')

test\_raw = pd.read\_csv('test.csv')

# Join data to analyse and process the set as one.

train\_raw['train'] = 1

test\_raw['train'] = 0

data = train\_raw.append(test\_raw, sort=False)

features = ['Age', 'Embarked', 'Fare', 'Parch', 'Pclass', 'Sex', 'SibSp']

target = 'Survived'

data = data[features + [target] + ['train']]

# Categorical values need to be transformed into numeric.

data['Sex'] = data['Sex'].replace(["female", "male"], [0, 1])

data['Embarked'] = data['Embarked'].replace(['S', 'C', 'Q'], [1, 2, 3])

train = data.query('train == 1')

test = data.query('train == 0')

# Drop missing values from the train set.

train.dropna(axis=0, inplace=True)

labels = train[target].values

train.drop(['train', target, 'Pclass'], axis=1, inplace=True)

test.drop(['train', target, 'Pclass'], axis=1, inplace=True)

from sklearn.model\_selection import train\_test\_split, cross\_validate

X\_train, X\_val, Y\_train, Y\_val = train\_test\_split(train, labels, test\_size=0.2, random\_state=1)

import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

from scipy.stats.stats import pearsonr

from sklearn.naive\_bayes import GaussianNB

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score, recall\_score, precision\_score, classification\_report, confusion\_matrix

%matplotlib inline

classifier = GaussianNB()

classifier.fit(X\_train, Y\_train)

y\_pred = classifier.predict(X\_val)

# Summary of the predictions made by the classifier

print(classification\_report(Y\_val, y\_pred))

print(confusion\_matrix(Y\_val, y\_pred))

# Accuracy score

from sklearn.metrics import accuracy\_score

print('accuracy is',accuracy\_score(Y\_val, y\_pred))

2. (Glass Dataset)

1. Implement Naïve Bayes method using scikit-learn library.

a. Use the glass dataset available in Link also provided in your assignment.

b. Use train\_test\_split to create the training and testing part.

2. Evaluate the model on testing part using score and classification\_report(y\_true, y\_pred)

1. Implement linear SVM method using scikit library

a. Use the glass dataset available in Link also provided in your assignment.

b. Use train\_test\_split to create the training and testing part.

2. Evaluate the model on testing part using score and classification\_report(y\_true, y\_pred)

Do at least two visualizations to describe or show correlations in the Glass Dataset.

Which algorithm got better accuracy? Can you justify why?

glass=pd.read\_csv("glass.csv")

glass.head()

features = ['Rl', 'Na', 'Mg', 'Al', 'Si', 'K', 'Ca', 'Ba', 'Fe']

target = 'Type'

X\_train, X\_val, Y\_train, Y\_val = train\_test\_split(glass[::-1], glass['Type'],test\_size=0.2, random\_state=1)

classifier = GaussianNB()

classifier.fit(X\_train, Y\_train)

y\_pred = classifier.predict(X\_val)

# Summary of the predictions made by the classifier

print(classification\_report(Y\_val, y\_pred))

print(confusion\_matrix(Y\_val, y\_pred))

# Accuracy score

from sklearn.metrics import accuracy\_score

print('accuracy is',accuracy\_score(Y\_val, y\_pred))

from sklearn.svm import SVC, LinearSVC

classifier = LinearSVC()

classifier.fit(X\_train, Y\_train)

y\_pred = classifier.predict(X\_val)

# Summary of the predictions made by the classifier

print(classification\_report(Y\_val, y\_pred))

print(confusion\_matrix(Y\_val, y\_pred))

# Accuracy score

from sklearn.metrics import accuracy\_score

print('accuracy is',accuracy\_score(Y\_val, y\_pred))

**Which algorithm you got better accuracy? Can you justify why?**

**We got better accuracy for NaÃƒÂ¯ve Bayes method which is 0.8372093023255814. Naive Bayes analysis works well with probabilistic concepts where as Linear SVM works better with linear regression logics. But to perform more accurately SVM requies large amounts of data to train and test the data. So, due to the amount of data Naive Bayes algorith gives better accuracy compared to Linear SVM.**

**https://github.com/NXI57230/Assignment1\_700725723**