**Machine Learning (Assignment # 3)**

**Name: Sindhu Rajanala**

**Student id: 700741228**

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

**Code:**

import pandas as pd

import seaborn as sns

from sklearn import preprocessing

import matplotlib.pyplot as plot

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

daf.head()

le = preprocessing.LabelEncoder()

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

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

**output:**

Graphical user interface, text, application

Description automatically generated

1. Do you think we should keep this feature?

Ans: Yes, the correlation between two element will be easy by using this feature.

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

**Code:**

daf.corr().style.background\_gradient(cmap="Greens")

**output:**

Graphical user interface, application, table

Description automatically generated

**Code:**

matrix = daf.corr()

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

plot.show()

**output:**

Text

Description automatically generated with medium confidence

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

**Code:**

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

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

train\_raw['train'] = 1

test\_raw['train'] = 0

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

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

target = 'Survived'

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

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

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

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

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

Scatter chart

Description automatically generated

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)

Text

Description automatically generated with low confidence

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 warnings

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

warnings.filterwarnings("ignore")

classifier = GaussianNB()

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

Graphical user interface, text, application

Description automatically generated

**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 training and testing part.**

**Code:**

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

glass.head()

Graphical user interface, text, application

Description automatically generated

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)

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

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

from sklearn.metrics import accuracy\_score

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

**output:**

Graphical user interface, text

Description automatically generated

**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 training and testing part.**

**Code:**

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

glass.head()

Graphical user interface, text, application

Description automatically generated

from sklearn.svm import SVC, LinearSVC

classifier = LinearSVC()

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

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

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

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

from sklearn.metrics import accuracy\_score

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

**output:**

A picture containing text

Description automatically generated

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

**Code:**

glass.corr().style.background\_gradient(cmap="Reds")

**output:**

A picture containing table

Description automatically generated

**Code:**

matrix = glass.corr()

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

plot.show()

**output:**

Graphical user interface, text, application, email

Description automatically generated

**Which algorithm you got better accuracy?**

**Ans:** The Naïve Bayes model and SVM are performing well on classifying spam messages with 98% accuracy but comparing the two models, **SVM is performing better**. These models can efficiently predict if the message is spam or not. **SVM is more powerful to address non-linear classification tasks**. SVM generalizes well in high dimensional spaces like those corresponding to texts.