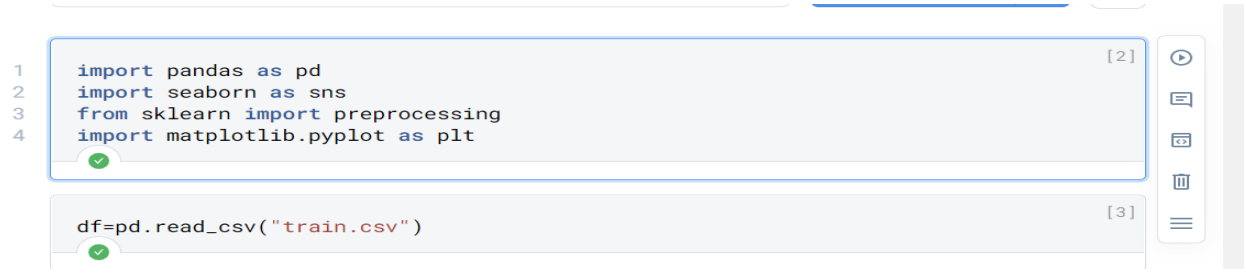
MACHINE LEARNING
ASSIGNMENT - 3

Name :Thandarupalli Naveen Goud
UCM ID :nxt46830
700# :700734683

Video Link:https://drive.google.com/file/d/1GM_F2PjUmeuWl8bZTr5FQYpKSY3FWD3l/view?usp=sharing
Github Link:
https://github.com/nxt46830/ML-Assignment-1

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.



```
[2]
import pandas as pd
import seaborn as sns
from sklearn import preprocessing
import matplotlib.pyplot as plt
```

```
[3]
df=pd.read_csv("train.csv")
```

```
df.head()
```

[4]

Visualize

	PassengerId int64	Survived int64	Pclass int64	Name object	Sex
0	1	0	3	Braund, Mr. Owen Harris	male
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	fema
2	3	1	3	Heikkinen, Miss. Laina	fema
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	fema
4	5	0	3	Allen, Mr. William Henry	male

[5]

```
le = preprocessing.LabelEncoder()
df['Sex'] = le.fit_transform(df.Sex.values)
df['Survived'].corr(df['Sex'])
```

```
-0.5433513806577553
```

[6]

```
matrix = df.corr()
print(matrix)
```

```
             PassengerId  Survived    Pclass       Sex       Age     SibSp  \
PassengerId     1.000000 -0.005007 -0.035144  0.042939  0.036847 -0.057527
Survived       -0.005007  1.000000 -0.338481 -0.543351 -0.077221 -0.035322
Pclass         -0.035144 -0.338481  1.000000  0.131900 -0.369226  0.083081
Sex             0.042939 -0.543351  0.131900  1.000000  0.093254 -0.114631
Age             0.036847 -0.077221 -0.369226  0.093254  1.000000 -0.308247
SibSp          -0.057527 -0.035322  0.083081 -0.114631 -0.308247  1.000000
Parch          -0.001652  0.081629  0.018443 -0.245489 -0.189119  0.414838
Fare            0.012658  0.257307 -0.549500 -0.182333  0.096067  0.159651
```

	Parch	Fare
PassengerId	-0.001652	0.012658
Survived	0.081629	0.257307
Pclass	0.018443	-0.549500
Sex	-0.245489	-0.182333
Age	-0.189119	0.096067
SibSp	0.414838	0.159651
Parch	1.000000	0.216225
Fare	0.216225	1.000000

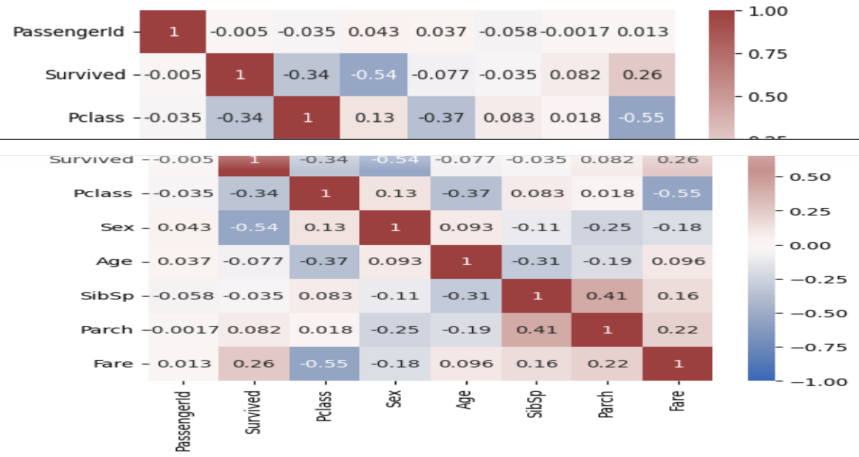
[7]

```
df.corr().style.background_gradient(cmap="Greens")
```

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Far
PassengerId	1.000000	-0.005007	-0.035144	0.042939	0.036847	-0.057527	-0.001652	0.0
Survived	-0.005007	1.000000	-0.338481	-0.543351	-0.077221	-0.035322	0.081629	0.2
Pclass	-0.035144	-0.338481	1.000000	0.131900	-0.369226	0.083081	0.018443	-0.
Sex	0.042939	-0.543351	0.131900	1.000000	0.093254	-0.114631	-0.245489	-0.
Age	0.036847	-0.077221	-0.369226	0.093254	1.000000	-0.308247	-0.189119	0.0
SibSp	-0.057527	-0.035322	0.083081	-0.114631	-0.308247	1.000000	0.414838	0.1
Parch	-0.001652	0.081629	0.018443	-0.245489	-0.189119	0.414838	1.000000	0.2
Fare	0.012658	0.257307	-0.549500	-0.182333	0.096067	0.159651	0.216225	1.0

[8]

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



[10]

```
# 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)
```

[11]

```
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)
```

[12]

```
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_m

%matplotlib inline
# Suppress warnings
warnings.filterwarnings("ignore")
```

[13]

```
classifier = GaussianNB()

classifier.fit(X_train, Y_train)
```

```
GaussianNB()
```

[14]

```
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))
```

```
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))
```

```
              precision    recall  f1-score   support

         0.0       0.79      0.80      0.80        85
         1.0       0.70      0.69      0.70        58

    accuracy                           0.76       143
   macro avg       0.75      0.74      0.75       143
weighted avg       0.75      0.76      0.75       143

[[68 17]
 [18 40]]
accuracy is 0.7552447552447552
```

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.

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 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.

```
[15]
glass=pd.read_csv("glass.csv")
```

```
[16]
glass.head()
```

Visualize

	RI float64	Na float64	Mg float64	Al float64	Si float64	K float64	Ca float64
0	1.52101	13.64	4.49	1.1	71.78	0.06	
1	1.51761	13.89	3.6	1.36	72.73	0.48	
2	1.51618	13.53	3.55	1.54	72.99	0.39	
3	1.51766	13.21	3.69	1.29	72.61	0.57	
4	1.51742	13.27	3.62	1.24	73.08	0.55	

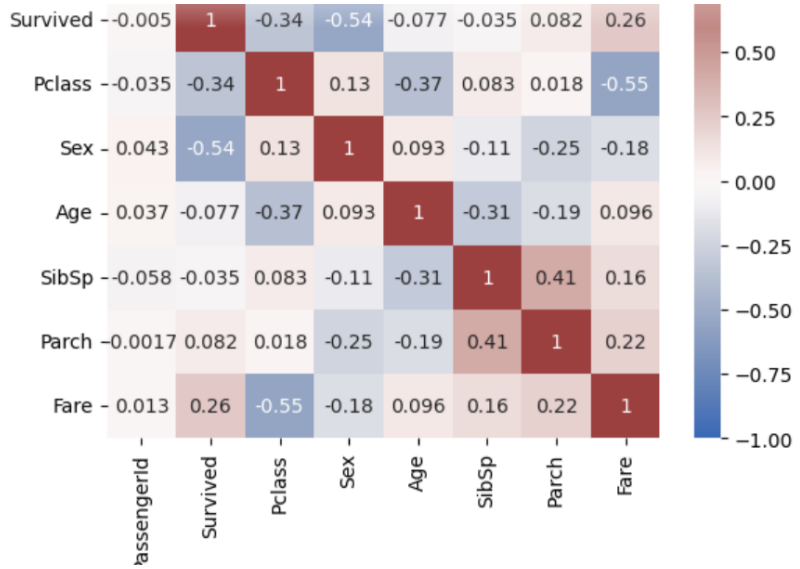
5 rows, showing 10 per page Page 1 of 1

```
[17]
glass.corr().style.background_gradient(cmap="Greens")
```

	RI	Na	Mg	Al	Si	K	Ca	Ba
RI	1.000000	-0.191885	-0.122274	-0.407326	-0.542052	-0.289833	0.810403	-0.000386
Na	-0.191885	1.000000	-0.273732	0.156794	-0.069809	-0.266087	-0.275442	0.326603
Mg	-0.122274	-0.273732	1.000000	-0.481799	-0.165927	0.005396	-0.443750	-0.492262
Al	-0.407326	0.156794	-0.481799	1.000000	-0.005524	0.325958	-0.259592	0.479404
Si	-0.542052	-0.069809	-0.165927	-0.005524	1.000000	-0.193331	-0.208732	-0.102151
K	-0.289833	-0.266087	0.005396	0.325958	-0.193331	1.000000	-0.317836	-0.042618
Ca	0.810403	-0.275442	-0.443750	-0.259592	-0.208732	-0.317836	1.000000	-0.112841
Ba	-0.000386	0.326603	-0.492262	0.479404	-0.102151	-0.042618	-0.112841	1.000000
Fe	0.143010	-0.241346	0.083060	-0.074402	-0.094201	-0.007719	0.124968	-0.058692
Type	-0.164237	0.502898	-0.744993	0.598829	0.151565	-0.010054	0.000952	0.575161

[18]

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



[19]

```
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))
```

```
              precision    recall  f1-score   support

           1       0.90      0.95      0.92        19
           2       0.92      0.92      0.92        12
           3       1.00      0.50      0.67         6
           5       0.00      0.00      0.00         1
           6       1.00      1.00      1.00         1
           7       0.75      0.75      0.75         4

    accuracy                           0.84        43
   macro avg       0.76      0.69      0.71        43
weighted avg       0.89      0.84      0.85        43

[[18  1  0  0  0  0]
 [ 1 11  0  0  0  0]
 [ 1  0  3  2  0  0]
 [ 0  0  0  0  0  1]
 [ 0  0  0  0  1  0]
 [ 0  0  0  1  0  3]]
accuracy is 0.8372093023255814
```

[20]

```
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))
```

```
              precision    recall  f1-score   support

           1       0.82      0.95      0.88        19
           2       1.00      0.50      0.67        12
           3       0.43      0.50      0.46         6
           5       0.00      0.00      0.00         1
           6       0.20      1.00      0.33         1
           7       0.00      0.00      0.00         4

    accuracy                           0.65        43
   macro avg       0.41      0.49      0.39        43
weighted avg       0.71      0.65      0.65        43

[[18  0  1  0  0  0]
 [ 4  6  0  1  1  0]
 [ 0  0  3  2  1  0]
 [ 0  0  1  0  0  0]
 [ 0  0  0  0  1  0]
 [ 0  0  2  0  2  0]]
accuracy is 0.6511627906976745
```

Reason:

- Which algorithm you got better accuracy? Can you justify why?
- Naive Bayes algorithm got better accuracy. Naive Bayes requires a small amount of training data, it tends to perform well for problems like spam detection and text classification.
- SVM algorithm typically don’t output easily interpretable probabilities and also svm is more expensive than naive bayes.
- Naive Bayes has better accuracy than SVM.