INDEX

NAME: Pragnya B S STD.: _____ SEC.: C ROLL NO.: _____ SUB.: ML Lab

S. No.	Date	Title	Page No.	Teacher's Sign / Remarks
1.	21/3/24	Import and export of Pandas Library		
2&3	29/3/24	End to End ML Project		
4	18/4/24	Linear Regression		
		Multiple Linear Regression		
5.	25/4/24.	Decision Tree - ID3		
6.	9/5/24	Logistic Regression		
		KNN classification.		
7.	23/5/24	K-means implementation		
		SVM		
		PCA		
8	30/5/24	Random forest ensemble method		
		Boosting.		

30/5/24

Pragnya
on
computer

Week 1

21/3/24

1) Python program for import and export of Pandas Library.

```
import pandas as pd
import:
url = "https://.../iris.data"
col_names = ["sepal_length_in_cm", "sepal_width_in_cm",
             "petal_length_in_cm", "petal_width_in_cm", "class"]
iris_data = pd.read_csv(url, names = col_names)
iris_data.head()
export:
iris_data.to_csv("cleaned_iris_data.csv")
```

Output:

sepal_length_in_cm	sepal_width_in_cm	petal_length_in_cm	petal_width_in_cm	class
5.1	3.5	1.4	0.2	Iris-setosa

Pandas

Lessons

1 Creating, Reading and Writing
2 Indexing, Selecting and Assigning
3 Summary Functions and Maps
4. Grouping and Sorting
5 Data Types and Missing Values
6 Renaming and Combining

Completed the Kaggle Pandas Certificate.

NLP
9/5/24

Week-2

29/3/24

End to End Machine Learning Project

1. Problem formulation
   Formulate the problem by considering what is the expected output and the features available.
2. Get the data
   * import OS
     - provides functions for interacting with OS
   * import urllib
     - provides functions for fetching URLs
   * import pandas as pd
     - provides functions to work with data sets
   * housing.head()
     - returns the top 5 rows of the file
   * housing.info()
     - gives the count of non-null values in each attribute
   * housing.describe()
     - gives the statistical measures such as count, mean, std, min, 25%, 50%, 75%, max for each attribute/features.
   * import matplot.pyplot as plt
     - a collection of command style functions which works like MATLAB
   * housing.hist(bins=50, figsize=(20,15))
     - histogram is plotted which is used for univariate analy and bins are used to specify the width of each group.
   * split_train_test()
     - It divides the dataset into train set and test set. test_si can also be specified to divide the data set
   * train_set.shape
     - Gives the shape of the train set (i.e, no. of rows & column
   * housing['id'].value_counts()
     - gives the count of values.

Week 3

4/4/24

Discover and visualize the data to gain insights.

* Visualize the data using matplotlib and seaborn libraries
* Calculating the standard correlation co-efficient of every pair of columns.

Prepare the data for Machine learning algorithm.

* Data cleaning, handling text and categorical data, custom transformers, feature scaling, transformation pipelines etc are done here.

Select and train model

* At first, linear regression model is used to train but the model is overfitting the data.
* To tackle this, decision tree Regressor model is used as it is capable of finding non-linear relationships within the data But the decision tree model is also overfitting so badly that it performs worse than the linear regression model.
* At last Random forest regressor is used. It is much better.

Fine tune your model.

At last, fine tuning of model is carried out, evaluating on the test set and then launch, monitor and maintain the system.

9/5/24

Week 4

→ python implementation of linear regression.

```
import numpy as np
import matplotlib.pyplot as plt

def estimate_coef(x, y):
    n = np.size(x)
    m_x = np.mean(x)
    m_y = np.mean(y)
    ss_xy = np.sum(y*x) - n*m_y*m_x
    ss_xx = np.sum(x*x) - n*m_x*m_x

    b_1 = ss_xy/ss_xx
    b_0 = m_y - b_1*m_x
    return (b_0, b_1)

def plot_regression_line(x, y, b):
    plt.scatter(x, y, color = "m", marker = "o", s = 30)
    y_pred = b[0] + b[1]*x
    plt.plot(x, y_pred, color = "g")
    plt.xlabel('x')
    plt.ylabel('y')

def main():
    x = np.array([0, 1, 2, ...., 9])
    y = np.array([1, 3, 2 .... 12])
    b = estimate_coef(x, y)
    print(b)
    plot_regression_line(x, y, b)
```

Output:

(b_0, b_1) = (1.2363..., 1.1696...)

→ Multiple linear regression:

```
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets, linear_model, metrics
data_url = "http://lib.stat.cmu.edu/datasets/boston"
raw_df = pd.read_csv(data_url, sep="\s+", skiprows=22, header=None)
X = np.hstack([raw_df.value[::2, :], raw_df.values
    [1::2, :2]])
y = raw_df.values[1::2, 2]
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                   test_size=0.4,
  random_state=1)
reg = linear_model.LinearRegression()
reg.fit(X=train, y=train)
print("Coefficients: ", reg.coef_)
print("variance score: {}".format(reg.score(X_test, y_test)))
plt.style.use('fivethirtyeight')
plt.scatter(reg.predict(X_train), reg.predict(X_train)
-y_train, color="green", s=10, label="Train data")
plt.scatter(reg.predict(X_test),
            reg.predict(X_test) - y_test,
            color="blue", s=10,
            label="Test data")
plt.hlines(y=0, xmin=0, xmax=50, linewidth=2)
plt.legend(loc='upper right')
plt.title("Residual Errors")
plt.show()
```

NL 9/5/24

Week-5

Decision tree - ID3.

25/4/24

```
import numpy as np
import pandas as pd
eps = np.finfo(float).eps
from numpy import log2 as log

url= "https://archive.ics.uci.edu/ml/machine-learning-databases/
     iris/iris.data"

df=pd.read_csv(url, header=None, names=['sepal length(cm)',
  'sepal width(cm)', 'petal length(cm)', 'petal width(cm)', 'species
print(f'Rows: {df.shape[0]}, Columns:{df.shape[1]}')
print(df.columns)

df.info()
df.describe()

def find_entropy(df):
target = df.keys()[-1]
entropy=0
values = df[target].unique()
for value in values:
    fraction= df[target].value_counts()[value]/len(df[targ
    entropy+= -fraction * np.log2(fraction)
return entropy

def average_information(df, attribute):
  target = df.keys()[-1]
  variables = df[attribute].unique()
  entropy2=0
  for variable in variables:
    entropy=0
    for target_variable in target_variables:
      num=len(df[attribute][df[attribute]==variable])
```

```
def find_winner(df):
    IG=[]
    for key in df.keys()[:-1]:
    IG.append(find_entropy(df)-average_information(df,key))
    return df.keys()[:-1][np.argmax(IG)]

def buildTree(df,tree=None):
    target= df.keys()[-1]
    attValue = np.unique(df[node])
    for value in attValue:
        subtable=get_subtable(df,node,value)
        clValue,counts= np.unique(subtable[target],return_count
                                  =True),

    if len(counts)==1:
        tree[node][value]=clValue[0]

    else:
        tree[node][value]=buildTree(subtable)

    return tree

tree=buildTree(df)

import pprint
pprint.pprint(tree).
```

NLP
9/5/24

Week-6

9/5/24

1. Logistic Regression Model:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns.

from sklearn.datasets import load_diabetes
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression.

diabetes = load_diabetes()
X, y = diabetes.data, diabetes.target
y_binary = (y > np.median(y)).astype(int)

X_train, X_test, y_train, y_test = train_test_split(
    X, y_binary, test_size=0.2, random_state=42)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train),
X_test = scaler.transform(X_test)

model = LogisticRegression()
model.fit(X_train, y_train)
```

Output: LogisticRegression()

```
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy: {:.2f}%".format(accuracy*100))
```

Accuracy: 73.03%

~~print~~

```
plt.figure(figsize=(8,6))
sns.scatterplot(x=X_test[:,2], y=X_test[:,8], hue=y_test,
    palette={0:'blue', 1:'red'}, marker='o')
plt.xlabel("BMI")
plt.ylabel("Age")
plt.title("Logistic Regression\n Accuracy: {:.2f}%".format(
    accuracy*100))
plt.legend(title="
```

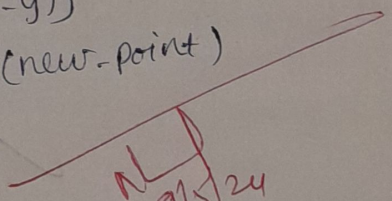
2. KNN classification model.

```
import matplotlib.pyplot as plt
from sklearn.neighbours import KNeighboursClassifier.

x=[4,5,10,4,3,11,14,8,10,12]
y=[21,19,24,17,16,25,24,22,21,21]
classes=[0,0,1,0,0,1,1,0,1,1]

data=list(zip(x,y))
print(data)
```

output: [(4,21),(5,19),(10,24),(4,17),(3,16),(11,25),(14,24),(8,22),
(10,21),(12,21)]

```
knn=KNeighboursClassifier(n_neighbours=1)
knn.fit(data,classes)

new_x=8
new_y=21
new_point=[(new_x,new_y)]
prediction=knn.predict(new_point)
print(prediction)
```

output: [0]




```
plt.scatter(x+[new_x],y+[new_y], c=classes+[prediction[0]])
plt.text(x=new_x-1.7, y=new_y-0.7, s=f"new_point,
  class:{prediction[0]}")

plt.show()

knn=KNeighboursClassifier(n_neighbours=5)
knn.fit(data,classes)
prediction=knn.predict(new_point)
print(prediction)
```

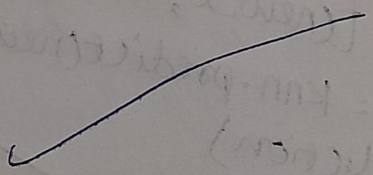
output: [1]

```
plt.scatter(x+[new_x],y+[new_y],c=classes+[prediction[0]])
plt.text(x=new_x-1.7, y=new_y-0.7, s=f"new Point, class:
    {prediction[0]}")

plt.show()
```

Week-4

K-means implementation

```
import pandas as pd
data = pd.read_csv("Iris.csv")
data.head()

import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.cluster import KMeans

x, y = load_iris(return_x_y = True)

kmeans = KMeans(n_clusters=3, random_state=2)
kmeans.fit(x)

pred = kmeans.fit_predict(x)
pred
```



30/5/24

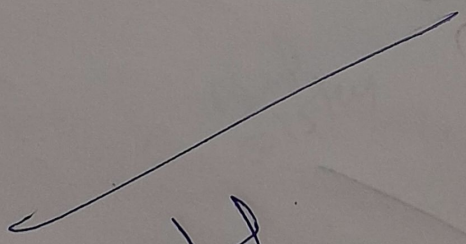
SVM:

```
from sklearn.datasets
import load_breastcancer
import matplotlib.pyplot as plt
from sklearn.inspection import DecisionBoundaryDisplay
from sklearn.svm import svc

cancer = load_breast_cancer()
  X = cancer.data[:, :2]
  y = cancer.target

svm = SVM svc(kernel="rbf", gamma=0.5, c=1.0)
svm.fit= (X,Y)

DecisionBoundaryDisplay.from_estimator(
    svm,
    X,
    response_method = "predict",
    cmap = plt.cm.spectral,
    alpha= 0.8,
    xlabel = cancer.feature_name[0],
    ylabel = cancer.feature_names[1],
)
plt.scatter( X[:,0], X[:,1], c=y, s=20, edgecolor:"k")
plt.show()
```

NLP
30/5/24

PCA

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.decomposition import PCA
from sklearn.preprocessing import standardscalar

from sklearn.datasets import load_breast_cancer
data= load_breast_cancer()
data.keys()

print(data['feature_name'])

df1= pd.DataFrame(data['data'], columns= data['feature_names'])

scaling = StandardScaler()
scaling.fit(df1)
scaled_data= scaling.transform(df1)
principal = PCA(n_components =3)
principal.fit(scaled_data)
X= principal.transform(scaled_data)
```



30/12/24

Week 8

30/5/2024

Random forest ensemble method:

```
import pandas as pd
import sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier.
from sklearn.metrics import accuracy_score, classification_report
import warnings
warnings.filterwarnings('ignore')

url = "https://raw.githubusercontent.com/datasciencedojo/
      datasets/master/titanic.csv"

titanic_data=pd.read_csv(url)
titanic_data = titanic_data.dropna(subset=['Survived']).
X = titanic_data[['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare']]
y = titanic_data['Survived']

X.loc[:, 'Sex'] = X['Sex'].map({'female': 0, 'male': 1}).
X.loc[:, 'Age'] = X['Age'].fillna(X['Age'].median(), inplace=True)
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.2, random_state=42)
rf_classifier = RandomForestClassifier(n_estimators=100,
                                       random=42)

rf_classifier.fit(X_train, y_train)
y_pred = rf_classifier.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
print("\nClassification Report:\n", classification_rep)
```

Boosting

```
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.ensemble import AdaBoostClassifier
import numpy as np
import matplotlib.pyplot as plt

x, y = load_breast_cancer(return_X_y = True)

x_train, x_test, y_train, y_test = train_test_split(x, y,
    test_size = 0.3, random_state = 23)

dtree = DecisionTreeClassifier(max_depth = 1, random_state = 23)
dtree.fit(x_train, y_train)
dt_pred = dtree.predict(x_test)

dt_acc = round(accuracy_score(y_test, dt_pred), 3)
print(f"Decision Tree Classifier Accuracy Score:", dt_acc)

ada = AdaBoostClassifier(n_estimators = 50, learning_rate = 0
ada.fit(x_train, y_train)
ada_pred = ada.predict(x_test)

ada_acc = round(accuracy_score(y_test, ada_pred), 3)
print(f"Decision Tree AdaBoost Model Accuracy Score:"
    ada_acc)
```