1BM21CS247

INDEX

Name Vyleri Kezheke Siddharth Class 6th sem

Roll No. _____ Subject ML LAB College _____

Sl. No.	Date	Title	Page No.	Teacher Sign / Remarks
1.	21/3/24	Python Program to import and export data using pandas library functions.		
2.	21/3/24	Exporting data frame to csv file		
3.	28/3/24	End to End ML Project		
4.	4/4/24	Simple & Multiple Linear Regression		
5.	18/4/24	Decision Tree		
6.	25/4/24	KNN & SVM		
7.	23/5/24	WEEK-7 - ANN & Random Forest		
8.	30/5/24	WEEK-8 - K Means & PCA		

Ans-3-24

DATE: 21/3/24 PAGE:

LAB - 1

Write a python program to import and export data using pandas library functions.

1) Download listings - austin .csv file.

Code :-

```
import pandas as pd
'air bnb - data = pd read - csv ("data/ listings _ austin. csv")
air - bnb - data. head ().
```

Output :-

	id	name	host - id	host - name
0	2265	Zen - East in the Heart of Aust	2466	Paddy
1	5245	Eco friendly, Colorful, Clean	2466	Paddy.
2.	5456	Walk to 6th, Rainey st and Co	8028	Sylvia
3.	5769	NW Austin Room	8186	Elizabeth.
4.	6413	Cem of a Studio near Down	13879	Todd.

2) Reading data from URL

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

# Define the column names
Col - names = [ "sepal length - in - cm", "sepal - width - in - cm",
        " petal - length - in - cm", "petal - width - in - cm",
        "class "]

# Read data from URL
iris - data = pd. read - csv ( url, names = Col - names) .
```

IP

DATE: PAGE:

iris - data. head().

Output :-

	sepal length(cm)	sepal - width(cm)	petal - length(cm)	petal. width(cm)	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa.

Kaggle certification (Pandas).

i) Creating, Reading and writing

- import pandas as pd.

Creating data:

i) pd.DataFrame ({'Yes': [50, 21], 'No': [131, 2]})

	Yes	No
0	50	131
1	21	2

ii) pd.DataFrame ({'Bob': ['I liked it', 'It was awful'], 'Sue': ['Pretty good', 'Bland']}, index = ['Product A', 'Product B'])

	Bob	Sue
Product A	I liked it	Pretty good
Product B	It was awful	Bland.

DATE: PAGE:

Reading Datasets :-

1) wine_reviews = pd.read_csv("../input/.../winemag.csv")

2) wine_reviews.shape
(129971, 14)

3) wine_reviews.head()

28/3/24

DATE: 28/3/24 PAGE:

WEEK - 2

End-End ML Project

1. Get the data:-

```
Import os
import tarfile
import urllib

DOWNLOAD_ROOT = "https://raw.githubusercontent.com/ageron/
                handson-ml2/master/"
HOUSING_PATH = os.path.join("data", "01")
HOUSING_URL = DOWNLOAD_ROOT + "datasets/housing/housing.tgz"

def fetch_housing_data(housing_url = HOUSING_URL,
                       housing_path = HOUSING_PATH):
    os.makedirs(name=housing_path, exist_ok=True)
    tgz_path = os.path.join(housing_path, "housing.tgz")
    urllib.request.urlretrieve(url=housing_url,
                               filename=tgz_path)
    housing_tgz = tarfile.open(name=tgz_path)
    housing_tgz.extractall(path=housing_path)
    housing_tgz.close()

# Download the data and call the function
fetch_housing_data()
```

DATE: PAGE:

2. Discover & Visualise the data to gain insights.

latitude / longitude info can be represented using .plot()

```
housing.plot(kind='scatter', x='longitude', y='latitude')
plt.show()
```

To estimate densities, use 'alph' parameter

```
housing.plot(kind='scatter', x='longitude', y='latitude', alpha=0.1)
plt.show()
```

To check the relation between two or more data info, 'corr' method is used.

```
housing[['population', 'median_house_value']].corr().
```

This generates a weak correlation b/w the specified parameters.

For a better correlation, we need to do

```
corr_matrix = housing.corr()

corr_matrix['median_house_value'].sort_values(
                                    ascending=False)
```

DATE: PAGE:

3. Prepare data for ML Algorithms.

```
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(strategy='median')

# This can handle only numbers
housing_num = housing.drop("ocean_proximity", axis=1)

# Now we fit imputer over our data.
# fitting is basically training.

imputer.fit(housing_num)

# Now we can use this to transform the
numbers by replacing their missing values
with corresponding medians.

X = imputer.transform(housing_num)

# X is a numpy array with the transformed
features. We add it back to the dataframe.

housing_tr = pd.DataFrame(data=X, index=housing_num
                          .index,
                          columns=housing_num
                          .columns).
```

DATE: PAGE:

4. Select & Train a Model :-

```
from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(X=housing_prepared, y=housing_labels)
some_data = housing.iloc[:5]    // Let's take small
some_labels = housing_labels.iloc[:5]    amount of the data
some_data_prepared = full_pipeline.transform(some_data)
lin_reg.predict(some_data_prepared)
```

some_labels.to_list() is not accurate, so we use another method.

```
from sklearn.metrics import mean_squared_error
housing_predictions = lin_reg.predict(housing_prepared)
lin_mse = mean_squared_error(housing_labels, housing_predictions)
lin_rmse = np.sqrt(lin_mse)
lin_rmse # this is the root mean squared error which is more accurate.
```

Linear regression has better performance but lower accuracy when compared to Decision tree & Random forest.
Linear regression is represented as $y = mx + c$
// Random Forest is a combination of multiple decision trees so it is better at capturing non-linear relationships.

Simple Linear Regression

RA1/4/24

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from pandas.core.common import random_state
from sklearn.linear_model import LinearRegression

df_sal = pd.read_csv('/content/Salary_Data.csv')
df_sal.head()
df_sal.describe()
plt.title('Salary Distribution Plot')
sns.distplot(df_sal['Salary'])
plt.show()

plt.scatter(df_sal['YearsExperience'], df_sal['Salary'],
            color='lightcoral')
plt.title('Salary vs Experience')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.box(False)
plt.show()

Split data
x = df_sal.iloc[:,:1]
y = df_sal.iloc[:,1:]
```

DATE: PAGE:

Split into Train test sets

```
X_train, X_test, y_train, y_test = train_test_split
                (X, y, test_size = 0.2,
                random_state = 0)
```

Train model

```
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

Predict results

```
y_pred_test = regressor.predict(X_test)
y_pred_train = regressor.predict(X_train)
```

Visualize predictions

```
plt.scatter(X_train, y_train, color = 'lightcoral')
plt.plot(X_train, y_pred_train, color = 'firebrick')
plt.title('Salary vs Experience (Training set)')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.legend([ 'X_train / Pred(y_test)'
             'X_train / y_train' ],
    title = 'Sal/Exp', loc = 'best', facecolor = 'white')
plt.box(False)
plt.show()
```

```
plt.scatter(X_test, y_test, color = 'lightcoral')
plt.plot(X_train, y_pred_train, color = 'firebrick')
plt.title('Salary vs Experience (Test set)')
plt.xlabel('Years of Exp')
plt.ylabel('Salary')
plt.show()
```

coefficient and intercept

```
print(f'Coefficient : {regressor.coef_}')
print(f'Intercept : {regressor.intercept_}')
```

Coefficient : [[9312.57]]
Intercept : [26780.099]

Multiple Linear Regression

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

df_start = pd.read_csv('/content/startup.csv')
df_start.head()

df_start.describe()
```

Distribution

```
plt.title('Profit Distribution Plot')
sns.distplot(df_start['Profit'])
plt.show()
```

Relation between Profit and R&D spend

```
plt.scatter(df_start['R&D Spend'], df_start['Profit'],
            color='light coral')
plt.title('Profit vs R&D spend')
```

DATE: PAGE:

```
plt.xlabel ('R&D spend')
plt.ylabel ('Profit')
plt.box (False)
plt.show()
```

Split into Independent / Dependent variables.

```
X = df_start.iloc[:, :-1].values
y = df_start.iloc[:, -1].values
```

One-hot encoding

```
ct = ColumnTransformer(transformers = [('encoder', OneHotEncoder(),
                                        [3])], remainder = 'passthrough')
X = np.array(ct.fit_transform(X))
```

Split into Train / Test sets

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                   test_size = 0.2, random_state = 0)
```

Train model

```
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

Predict results

```
y_pred = regressor.predict(X_test)
```

Compare predictions

```
np.set_printoptions(precision = 2)
result = np.concatenate((y_pred.reshape(len(y_pred),1),
                         y_test.reshape(len(y_test),1)),1)
```

WEEK - 4

RA-1611/24

Decision Tree:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns.
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score

# Load iris dataset
iris_data = load_iris()
X = iris_data.data
y = iris_data.target

# Split dataset into training and testing sets.

X_train, X_test, y_train, y_test = train_test_split(X, y,
                                   test_size = 0.2,
                                   random_state = 42)

# initialize decision tree classifier
clf = DecisionTreeClassifier()

# fit classifier to the training data
clf.fit(X_train, y_train)
output:
        DecisionTreeClassifier()
y_pred = clf.predict(X_test)
```

DATE: PAGE:

```
# Calculate Accuracy:
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

# plot decision Tree
plt.figure(figsize=(12,8))
plot_tree(clf, filled=True, feature_names=iris.data.
          feature_names, class_names=iris.data.
          target_name);
plt.show()
```

output:

Accuracy 1.0

DATE: 25/4/24 PAGE:

WEEK-5

Build logistic regression model for given dataset

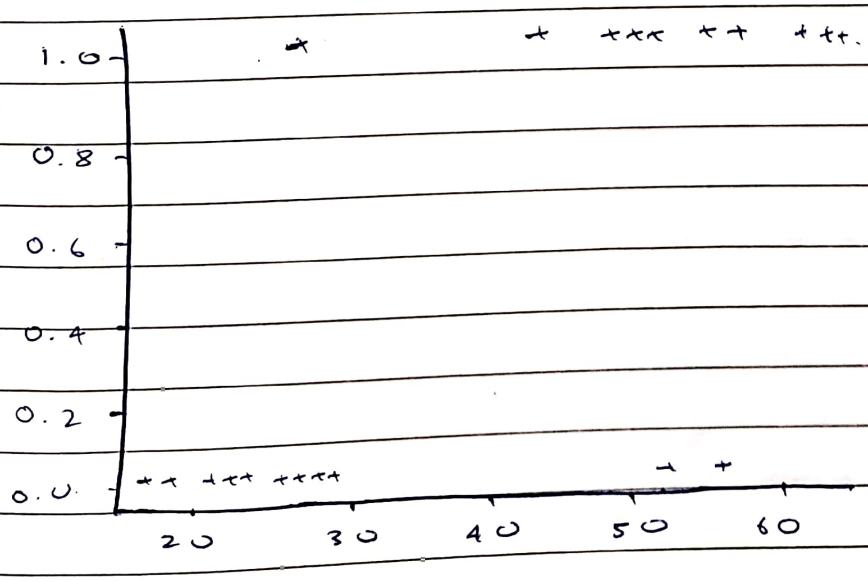
Code:-

```
import pandas as pd
from matplotlib import pyplot as plt.
%matplotlib inline
df = pd.read_csv("insurance_data.csv")
df.head().
```

OUTPUT:-

	age	bought_insurance
0	22	0
1	25	0
2	47	1
3	52	0
4	46	1

```
plt.scatter(df.age, df.bought_insurance, marker='+', color='red')
```



DATE: PAGE:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df[['age']],
df.bought_insurance, train_size=0.8)
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train, y_train)
y_predicted = model.predict(X_test)
model.predict_proba(X_test)
model.score(X_test, y_test)
```

OUTPUT:-

0.8333

```
print(y_predicted)
[0 1 0 0 0 0]
```

Aa/5/24 DATE: PAGE:

WEEK - 6

I) Build KNN Classification model for Iris dataset

1) Importing necessary libraries

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt.
```

2> load data

```
iris_data = pd.read_csv('Iris.csv')
```

3> display sample of dataset

```
iris_data.head()
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

4> Encoding the y values into numbers.

```
label_mapping = {'Iris-setosa':1, 'Iris-versicolor':2,
                 'Iris-virginica':3}
iris_data['Species'] = iris_data['Species'].map(label_mapping)
```

PTO

DATE: PAGE:

5> creating a scatter plot

```
plt.scatter(iris_data['Sepal Length (cm)'], iris_data['Sepal Width (cm)'],
    c = iris_data['species'], cmap = 'viridis')
plt.xlabel('Sepal Length (cm)')
plt.ylabel('Sepal Width (cm)')
plt.title('Scatter Plot of Iris Dataset (Sepal Features)')
plt.colorbar(label='species')
plt.show()
```

6> Splitting the dataset into X and y where X contains input features and y contains the target label.

```
iris_data.drop('Id', inplace=True, axis=1)
X = iris_data.drop('species', axis=1)
y = iris_data['species']
```

7> Splitting the X and y into X_test, X_train, y_train, y_test.

```
X_train, X_test, y_train, y_test = train_test_split
                                   (X, y, test_size=0.4,
                                    random_state=42)
```

8> Creating a KNN model

```
knn = KNeighborsClassifier(n_neighbors=5)
```

9> fitting the model

```
knn.fit(X, y)
```

DATE: PAGE:

```
y_pred = knn.predict(X_test)
acc = accuracy_score(y_pred, y_test)
print(acc)
-> 1.0
```

SVM:

```
from sklearn.svm import SVC
model = SVC()
model.fit(X, y)
y_pred = model.predict(X_test)
acc = accuracy_score(y_pred, y_test)
print(acc)
-> 1.0
```

DATE: PAGE:

WEEK - 7 23-5-24

ANN:

```
import numpy as np
X = np.array(([2,9],[1,5],[3,6]), dtype = float)
y = np.array(([92],[86],[89]), dtype = float)
X = X/np.amax(X, axis=0)
y = y/100
# Variable initialization
epoch = 5000
# Setting training iterations
# Setting learning rate
lr = 0.1
# input layer - neurons = 2
# number of features in data set
hiddenlayer - neurons = 3
# number of hidden layers neurons
output - neurons = 1
# number of neurons at output layer.
# weight and bias initialization
wh = np.random.uniform(size = (input layer - neurons,
        hidden layer - neurons))
bh = np.random.uniform(size = (1, hidden layer - neurons))
wout = np.random.uniform(size = (hiddenlayer - neurons,
        output - neurons)).
bout = np.random.uniform(size = (1, output - neurons))
```

DATE: PAGE:

```
# Sigmoid Function
def sigmoid(x):
    return 1/(1+np.exp(-x))

# Derivative of Sigmoid Function
def derivatives_sigmoid(x):
    return x*(1-x)

#draws a random range of numbers uniformly of
dim x*y for i in range(epoch):
# Forward Propagation
for i in range(epoch):
    hinpl=np.dot(X,wh)
    hinp = hinp+bh
    hlayer_act = sigmoid(hinp)
    outinp1 = np.dot(hlayer_act, wout)
    outinp = outinp1 + bout
    output = sigmoid(outinp)
# Backpropagation
    EO = y - output
    outgrad = derivatives_sigmoid(output)
    d_output = EO * outgrad
    EH = d_output.dot(wout.T)
    hiddengrad = derivatives_sigmoid(hlayer_act)
    d_hiddenlayer = EH * hiddengrad.
```

PTO

DATE: PAGE:

```
wout += hlayer1.T.dot(d_output) * lr
wh += X.T.dot(d_hiddenlayer) * lr
print("Input: \n" + str(X))
print("Actual output: \n" + str(y))
print("Predicted output: \n", output)
```

OUTPUT:-

```
[[0.6666667  1.       ]
 [0.3333333  0.5555556]
 [1.         0.6666667]]
```

Actual output:

```
[[0.92]
 [0.86]
 [0.89]]
```

Predicted output:

```
[[0.79197201]
 [0.77476931]
 [0.7964727 ]]
```

DATE: PAGE:

Random Forest

1> from sklearn.ensemble import RandomForestClassifier
import pandas as pd
from sklearn.metrics import accuracy_score

2> df = pd.read_csv("/content/drive")
melbourne_data = df.dropna(axis=0)
y = melbourne_data.Price
melbourne_features = ['Rooms', 'Bathroom', 'Landsize', 'BuildingArea', 'YearBuilt', 'Lattitude', 'Longitude']
X = melbourne_data[melbourne_features].
df.head().

3> from sklearn.model_selection import train_test_split

4> X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.8, random_state=0)
model = RandomForestClassifier().
model.fit(X, y).
y_predicted = model.predict(X_test).
accuracy = accuracy_score(y_predicted, y_test).
print(accuracy)
0.99

DATE: 30/5/24 PAGE:

WEEK - 8

K- means.

1> Imports :-

```
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.cluster import KMeans
import pandas as pd
import numpy as np
```

2> Load Dataset :-

```
iris = datasets.load_iris()
X = pd.DataFrame(iris.data)
X.columns = ['Sepal_Length', 'Sepal_Width', 'Petal_Length',
             'Petal_Width']
y = pd.DataFrame(iris.target)
y.columns = ['Targets']
```

3> K-Means Model

```
model = KMeans(n_clusters = 3)
model.fit(X)
```

4> Plot the figures :-

```
plt.figure(figsize=(14,14))
colormap = np.array(['red', 'lime', 'black'])
plt.subplot(2,2,1)
plt.scatter(X.Petal_Length, X.Petal_Width,
    c = colormap[y.Targets], s=40)
plt.title('Real Clusters')
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
```

DATE: PAGE:

```
#Plot the Models Classifications
plt.subplot(2,2,2)
plt.scatter(X.Petal_Length, X.Petal_Width, c = color map
 [model.labels_], s=40)
plt.title('K-Means Clustering')
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
```

PCA

1> Imports :

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

2> Import Dataset :-

```
from sklearn.datasets import load_breast_cancer
cancer = load_breast_cancer()
cancer.keys()
df = pd.DataFrame(cancer['data'], columns = cancer['feature_
                                                  names'])
```

3> PCA Visualization

```
from sklearn.preprocessing import StandardScaler.
scaler = StandardScaler()
scaler.fit(df)
Scaled_data = scaler.transform(df)
from sklearn.decomposition import PCA
pca = PCA(n_components =2)
 pca.fit(scaled_data)
```

DATE: PAGE:

```
X_pca = pca.transform(scaled_data)
scaled_data.shape
X_pca.shape
```

A> Plot the graph:

```
plt.figure(figsize=(8,6))
plt.scatter(X_pca[:,0], X_pca[:,1], c=cancer['target'],
    cmap='plasma')
plt.xlabel('First principal component')
plt.ylabel('Second Principal Component')
```