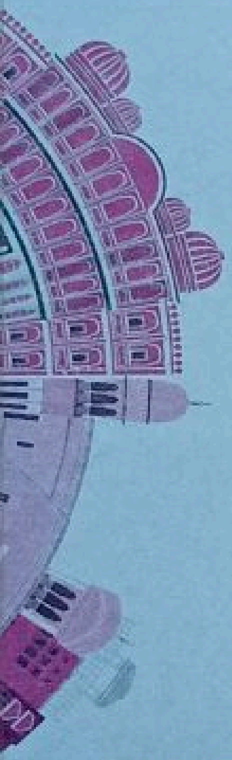

amily that once
the Old City, or
ately street grid
complex. With
ce.

INDEX

Name Swastik Srivastava Sub.

Std.: Div. Roll No. 18M21CS228

Telephone No. E-mail ID.

Blood Group. Birth Day.

Sr.No.	Title	Page No.	Sign./Remarks
1.	Python pgm to import		2/3
	and export data		
2.	~~exp to~~ ML Project		28/3
3.	Linear & multi-linear		18/4
	regression		
4.	Decision tree		25/4
5.	Logistic regression		
6	KNN Classification,		9/5
	SVM		
7	ANN, Randomforests		
8	Adaboost		23/5
~~8~~ 9	k-means		
10	Principal		30/5
	component analysis		

SURYA Gold

Date 21/3/24 Page

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

```
import pandas as pd

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

O/P

	sepal_length_in_cm	sepal_width_in_cm	petal_length_in_cm	petal_width_in_cm	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa

```
iris_data.to_csv("cleaned_iris_data.csv")

~~data1 = pd.read_csv("cleaned_iris_data.cs~~
data1 = pd.read_csv("Downloads/iris/iris.data")
```

21/3/24

SURYA Gold
Date 26/9/24 Page

2. Get The Data

→ fetch_housing_data()

```
import pandas as pd
def load_housing_data(housing_path=HOUSING_PATH):
    data_path = os.path.join(housing_path, "housing.csv")
    return pd.read_csv(data_path)
```

Quick look at the Dataset

```
housing = load_housing_data()
housing.head()
housing.info()

housing['ocean_proximity'].value_counts()
housing.describe()
```

→

```
import matplotlib.pyplot as plt
import seaborn as sns
housing.hist(bins=50, figsize=(20,15))
plt.show()
```

→

```
import numpy as np
def split_train_test(data, test_ratio=0.2):
    return data.iloc[train_indices], data.iloc[test_indices]

train_set, test_set = split_train_test(data=housing)
len(train_set) = 16512
len(test_set) = 4128
```

→ % of data to be explored

-> housing['income_cat'] = pd.cut(x= housing['median_income'], bins=[0, 1.5, 3, 4.5, 6, np.inf],
labels=[1,2,3,4,5])
housing['income_cat'].hist()

Here we generated test set & train set, which is an important part of ML.

3. Visualization

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

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

-> housing[['population','median_house_value']].corr()

-> corr_matrix = housing.corr()

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

from pandas.plotting import scatter_matrix
attributes = ['median_house_value', 'median_income',
'total_rooms', 'housing_median_age']

scatter_matrix(frame=housing[attributes],
figsize=(12, 8))
plt.show()

SURYA Gold

Date ____ Page ____

4. Preparing the Data for ML Algorithms

I Data cleaning

II Handling text & categorical attributes

III Custom Transformers

IV Feature Scaling

V Transformation pipelines

5. Select and train model

from sklearn.linear_model import LinearRegression

lin_reg = LinearRegression()

6. Fine Tune model

from sklearn.model_selection import GridSearchCV

final_model = grid_search.best_estimator_

28/3/24

SURYA Gold

Date______ Page______

~~LAB-~~

LAB- 3

implement Linear and Multi-linear regression algorithm using appropriate dataset

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

df_sal = pd.read_csv("salary_data.csv")
df_sal.head()
plt.title("salary plot")
sns.distplot(df_sal['salary'])
plt.show()

plt.scatter(df_sal['YearsExperience'],
df_sal['salary'], color = 'lightcoral')
plt.title('salary vs Experience')
plt.show
```

Split Data

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

split into Train and 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)
```

SURYA Gold

Date______ Page______

```
y-pred-test = regressor.predict (x-test)
y-pred-train = regressor. predict (x-train)
```

Visual Predictions

```
plt. scatter (x-train, y-train, color = 'lightcoral')
plt. plot (x-train, y-pred-train)
plt. show
```

Coefficient and Intercept

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

Coefficient : [[9312.57512]]
Intercept : [26780.0991]

Multiple Linear Regression

```
df. start = pd.read-csv ('startups.csv')
df-start.head ()
df. describe ()
```

Distribution

```
plt. title ('Profit Distribution Plot')
sns. distplot (df-start ['profit']
plt. show()
```

Relationship between Profit & R&D Spend

SURYA Gold

Date ______ Page ______

```
plt.scatter(df_start['R&D spend'], df_start
        ['profit'])
plt.show()
```

Split Data

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

split into train/test data

```
x_train, x_test, y_train, y_test = train_test_
split (x, y, test_size = 0.2, random_state = 0)
```

Train Model

```
regresor = LinearRegression()
regresor.fit (x_train, y_train)
```

Predicted Results

```
y_pred = regresor.predit (x_test)
```

Compare prediction

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

```
array([[103015.2, 103283.38],
       [103258.2.28, 144259.4],
       [132447.74, 146121.95],
      ])
```

18/4

SURYA Gold

Date 18/4/24 Page

6. Use an appropriate data set for building the decision tree (ID3) & apply this knowledge to classify a new sample.

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier,
plot_tree
import matplotlib.pyplot as plt

iris = load_iris()
 X = iris.data
 y = iris.target
 feature_names = iris.feature_names
 target_names = iris.target_names

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

clf = DecisionTreeClassifier(criterion = "entropy")
clf = DecisionTreeClassifier(criterion = 'Gini')

 clf.fit (X_train, y_test)

 y_pred = clf.predict(X_test)

 accuracy = clf.score(X_test, y_test)
 print("Accuracy:", accuracy)
```

SURYA Gold

Date ______ Page ______

```
plt. figure ( figsize = (12,8))
Plot-tree (clf, feature_names = feature_names,
            class_names = target_names, filled = True)

plt.show()
```

OUTPUT

Accuracy - 0.983

Petal length ≤ 2.45
gini = 0.665
samples = 90
value = [27, 31, 32]
class = virginica

gini = 0.0
samples = 27
value = [27, 0, 0]

Petal width ≤ 1.75
gini = 0.5
sample = 63
value = [0, 31, 32]
class = virginica

18/4/24

SURYA Gold

Lab-5

Date 25/4/24 Page

```
import pandas as pd
from ~~map~~ matplotlib import pyplot as plt
%matplotlib inline

df = pd.read_csv("/content/insurance_data/
                   insurance_data.csv")
df.head()

plt.scatter(df.age, df.bought_insurance, marker='+', color='red')
from sklearn.model_selection import train_test_split
X_train, X_test, 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)
print(y_predicted)
[0 1 1 0 1 0]

model.coef_
  arr([[0.1446]])

model.intercept_
  arr([-5.199])
```

SURYA Gold

Date________ Page________

```
import math
def sigmoid(x):
    return 1/(1+ math.exp(-x))


def prediction_function(age):
    z = 0.042 * age - 1.53
    y = sigmoid(z)
    return y

age = 35
prediction_function(age)
= 0.485

age = ~~20~~ 43
prediction_function(age)
= 0.5686
```

25/4/24

SURYA Gold

Date______ Page______

Lab- 6

Build KNN classification for given dataset

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbours import KNeighborsClassifier
from sklearn.metrics import accuracy score
import matplotlib.pyplot as plt

df = pd.read_csv(" diabetes2.csv")
df.head();

plt.scatter(df['Insulin'], df['BMI'], c=df['Outcome'],
            cmap='viridis')
plt.xlabel('feature1')
plt.ylabel('feature2')
plt.title('Scatter Plot of Data')
plt.show()

x = df.drop('Outcome', axis=1)
y = df['Outcome']

x_train, x_test, y_train, y_test = train_test_split(x, y,
                              test_size=0.45, random_state=42)

k = 10
knn_classifier = KNeighborsClassifier(n_neighbors=k)

knn_classifier.fit(x_train, y_train)
y_pred = knn_classifier.predict(x_test)

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

O/P 0.7283

SURYA Gold

Date______ Page______

SVM

```
import Pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt

df = pd.read_csv("Iris.csv")
print("Dataset")
print(df.head)

class_mapping = {'Iris-setosa':0, 'Iris-versicolor': 1,
                 'Iris-virginica': 2}
df['target_numeric'] = df['Species'].map(class_mapping)

plt.scatter(df['SepalLengthCm'], df['PetalLengthCm'],
            c= df['target_numeric'], cmap='viridis')
plt.xlabel('Sepal width (cm)')
plt.ylabel('Petal width (cm)')
plt.scatter('Scatter plot of Iris data')
plt.show()

X= df.drop('Species', axis=1)
y= df['Species']

x_train, x_test, y_train, y_test = train_test_split(x, y,
                           test_size =0.45, random_state =50)

svm_classifier = SVC(kernel ='linear')
svm_classifier.fit(x_train, y_train)
y_pred = svm_classifier.predict(x_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

O/P 0.9852

9/5/24

SURYA Gold

Date 23/5/24 Page

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

epoch = 5000
lr = 0.1
inputlayer_neurons = 2
hiddenlayer_neurons = 3
output_neurons = 1
wh = np.random.uniform(size=(inputlayer_neurons,
                hiddenlayer_neurons))
bh = np.random.uniform(size=(1, hiddenlayer_neurons))
wout = np.random.uniform(size=(hiddenlayer_neurons,
                output_neurons))
bout = np.random.uniform(size=(1, output_neurons))

def sigmoid(x):
    return 1/(1+np.exp(-x))

def derivates_sigmoid(x):
    return x*(1-x)

for i in range(epoch):
    hinp1 = np.dot(x, wh)
    hinp = hinp1 + bh
    hlayer_act = sigmoid(hinp)
    outinp1 = np.dot(hlayer_act, wout)
    outinp = outinp1 + bout
    output = sigmoid(outinp)
```

SURYA Gold

Date Page

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

wout += hlayer_act.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)

O/P
[[0.6667 1]
[0.333 0.5556]
[1. 0.6667]]

Actual O/P:
[[0.92]
[0.86]
[0.89]]

Predicted O/P
[[0.7782]
[0.7661]
[0.7734]]

SURYA Gold

Date 23/5/24 Page

Randomforest

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

data = pd.read_csv('Iris.csv')

x = data.drop('Species', axis=1)
y = data['Species']

x_train, x_test, y_train, y_test =
        train_test_split(x, y, test_size=0.39,
                         random_state=49)

rf_classifier = RandomForestClassifier(n_estimators=100,
                         random_state=42)
rf_classifier.fit (x_train, y_train)

y_pred = rf_classifier.predict (x_test)

accuracy = accuracy_score(y_test, y_pred)
print (f'Accuracy: {accuracy:.2f}')
```

O/P

Accuracy - 0.98

23/5/24

SURYA Gold

Date 23/5/24 Page

AdaBoost

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

df = pd.read_csv("/content/iris.csv")
df.head()
y = df['species']
x = df.drop(['species'], axis=1)

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split
    (x, y, test_size=0.3, random_state=0)

from sklearn.ensemble import AdaboostClassification
adb = Adaboost classifi~~cation~~ er()
adb_model = adb.fit (x_train, y_train)
y_pred = adb.predict (x_test)
y2 = adb.predict (x_train)

from sklearn.metrics import accuracy_score
acc_score = accuracy_score(y_pred, y_test)
print (accuracy_score)
```

O/P Accuracy = 0.977

23/5

Lab-10

SURYA Gold

Date 30/5/24 Page

k-means clustering algorithm

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

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

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

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.Target], s=40)
plt.title('Real Clusters')
plt.xlabel('Petal Length')
plt.ylabel('Petal width')

plt.subplot(2,2,2)
plt.scatter(X.Petal_Length, X.Petal_width,
            c=colormap[model.labels_],
            s=40)
```

plt.title ('K-Means Clustering')
plt.xlabel ('Petal Length')
plt.ylabel ('Petal Width')
plt.show()

PCA


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

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

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

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)
x_pca = pca.transform(scaled_data)

scaled_data.shape
x_pca.shape

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')
```

30/5/24