21/3/24.

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

(a) Importing a csv file using the read_csv() function

```
import pandas as pd
data = pd.read_csv("C:\Users\Admin\cleared_iris_data.csv")
data.head()
```

Output:

Unnamed:0	sepal_length_in_cm	sepal_width_in_cm	class
0			
1			
2			
3			
4			

(b) Reading data from URL.

```
import pandas as pd
url = "https://..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()
```

output:

	sepal_length_in_cm	sepal_width_in_cm	petal_length_in_cm	petal_width_in_cm	class
0					
1					
2					
3					
4					

④ Exporting the data frame to a csv file

```
iris_data_to_csv ("cleared_iris_data_csv")
```

output :

exports the file to the current working directory.

LAB-2 End to end project

1. Look at the big picture : Performance Measure.
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)
housing = load_housing_data()
housing.head()
housing.info()
housing.describe()
import matplotlib.pyplot as plt
import seaborn as sns
housing.hist(bins = 50, figsize = (20,15))
plt.show()
```

pd.cut

```
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()
```

3. Discover and visualize the Data to gain insights

```
strat_Train_set.shape, strat_Test_set.shape
```

Visualizing

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

Correlation

```
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']
scatter_matrix(frame = housing[attributes], figsize=(12,8))
plt.show()
housing['rooms_per_household'] = housing['total_rooms']
/housing['household']
```

(4) Prepare the data for Machine Learning Algorithms

```
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(strategy = 'median')
housing_num = housing.drop('ocean_proximity',
                           axis =1)
imputer.fit(housing_num)
imputer.statistics_
housing_num.median().values
```

- `from sklearn.preprocessing import OrdinalEncoder`
  `ordinal_encoder = OrdinalEncoder()`
  `ordinal_encoder.categories_`

- `attr_adder = CombinedAttributesAdder(add_bedroom_per_room = False)`

```
from sklearn.compose import column_transformation
num_attributes = having_num.columns.tolist()
```

(5) Select and Train a Model:

- from sklearn.linear_model import LinearRegression
  lin_reg = LinearRegression()
  lin_reg.fit(x = housing_prepared, y = housing_labels)

- housing_predictions = lin_reg.predict(housing_prepared)
  lin_mse = mean_squared_error(housing_label housing_predictions)
  lin_rmse = np.sqrt(lin_mse)
  lin_rmse

- tree_reg = DecisionTreeRegressor()
  tree_reg.fit(x = housing_prepared, y = housing_labels)

- forest_reg = RandomForestRegressor()
  forest_reg.fit(x = housing_prepared, y = housing_labels)

(6) Fine-tune your model:

- grid_search.best_params_
- grid_search.best_estimator_
  cat_encoder = full_pipeline.named_transformers_['cat']
- final_model = grid_search.best_estimator_
- X_test_prepared = full_pipeline.transform(x = X_test)
  final_rmse = np.sqrt(final_mse)
  final_rmse
- squared_errors = (y_test - final_predictions) ** 2

Launch, Monitor and Maintain your system.

30/5/24

LAB-3 Simple Linear Regression and 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 pandas.core.common import random_state
from sklearn.linear_model import LinearRegression
df_sal = pd.read_csv('c:\salary_data.csv')
df_sal.head()
df.sal.describe()

plt.title('salary Distribution Plot')
sns.distplot(df_sal['salary'])
plt.show()

x = df_sal.iloc[:, :1]
y = df_sal.iloc[:, 1:]
x_train, x_Test, y_train, y_Test = train_test_split(x, y, test_size = 0.2, random_state=0)
regressor = LinearRegression()
regressor.fit(x_train, y_train)
y_pred_test = regressor.predict(x_test)
y_pred_train = regressor.predict(x_train)
```

Steps:

① Import libraries
② Import data
③ Analyze data
④ Split data
⑤ Predict results
⑥ Visualize predictions

Multiple linear regression

① dt_start = pd.read_csv("/content/SD_start")
dt.start.head()

② plt.title("profit distribution plot")
sns.displot(dt_start['Profit'])
plt.show()

③ spitting
x = dt_start.iloc[:,:-1].values
y = dt_start.iloc[:,+1].values

④ x_train, x_test, y_train, y_test = train_train_split(x, y, test_size = 0.2, random_start = 0)

⑤ regressor = linear_regression()
regressor.slit(x_train, y_train)

⑥ Predict
y_pred = regressor.predict(x_test)

Steps:

1. Import libraries
2. Import data
3. Analyze data
4. Split into independent / dependent variables
5. Predict results
6. compare predictions

LAB-4

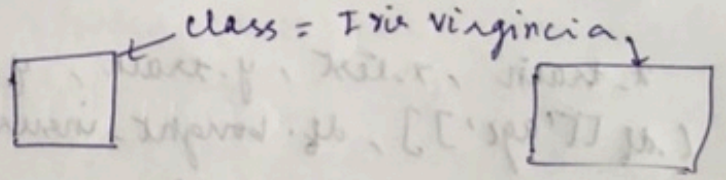
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import DecisionTreeClassifier
plot_tree
from sklearn.metrices import accuracy_score
import matplotlib.pyplot as plt

iris_data = pd.read_csv('Iris.csv')
iris_data.head()
iris_data.info()

<class 'pandas.core.frame.DataFrame'>.

iris_data.drop('Id', inplace=True, axis=1)
x = iris_data.drop('Species', axis=1)
y = iris_data['Species']

x_train, x_test, y_train, y_test = train_test_split
(x, y, test_size=0.4, random_state=42)

dt_classifier = DecisionTreeClassifier(criterion='entropy',
random_state=42)

dt_classifier.fit(x_train, y_train)

DecisionTreeClassifier

DecisionTreeClassifier(criterion='entropy', random_state=42
plt.figure(figsize(12,8))
plot_tree(dt_classifier, feature_names=x.columns,
          class_names=dt_classifier.classes_)
```

filled: True)
plt.show( )

petal length cm <= 2.45
entropy = 1.581
samples = 90
value = [27, 31, 32]
class = Iris-Virginica

entropy = 0.0
samples = 27
value = [27, 0, 0]
class = Iris-Setosa

petal width cm <= 1.75
entropy = 1.0
samples = 63
value = [0, 31, 32]




```
y_pred = dt_classifier.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy: {accuracy}")
```

Accuracy: 0.9833.

Assy 3015

WEEK-5

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

from sklearn.metrices import accuracy_score
df = pd.read_csv("/content/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_train, y_test = train_test_split
(df[['age']], df.bought_insurance, train_size=0.6,
    random_state=0)

from sklearn.linear_model import logistic regression
model = logistic Regression()
model.fit(x_train, y_train)
y_predicted = model.predict(x_test)
model.predict_proba(x_test)
y_predicted
array([1,1,1,1,1,0,0,0,0,0,0])

accuracy = accuracy_score(y_predicted, y_test)
print(accuracy)
0.8181818182

model.intercept_
array[-6.19091564])

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

age = 35
prediction - function (age)
0.485004499 99
prediction - function (age)
0. 56857705

age = 86
prediction - function (age)
0. 889141 3248300

SVM - Iris Dataset

9/5/24

```
iris = pd.read_csv('IRIS.csv')
sns.pairplot(data = iris, hue = 'species', palette =('set2'))
from sklearn.model_selection import train_test_split
x = iris.iloc[:, :-1]
y = iris.loc[:, 4]
x_train, x_test, y_train, y_test = train_test_split
    (x, y, test_size = 0.30)
from sklearn.svm import svc
model = svc()
model.fit(x_train, y_train)
    svc()
model2 = svc(C=5, kernel = 'rbf')
model2.fit(x_train, y_train)
pred = model2.predict(x_test)
from sklearn.metrices import classification_report,
confusion_matrix
print(confusion_matrix(y_test, pred))
print(classification_report(y_test_pred))
```

KNN

```
classifier = KNeighborclassifier(n_neighbours = 3)
classifier.fit(x_train, y_train)
y_pred = classifier.predict(x_test)
cm = confusion_matrix(y_test, y_pred)
accuracy = accuracy_score(y_test, y_pred) * 100
print('Accuracy of our model is equal' +
      str(round(accuracy, 2)) + '%')
```

Accuracy of our model = 93.33%

Cross validations

```
k_list = list(range(1,50,2))
cv_scores = []

for k in k_list:
knn = KNeighborsClassifier(n_neighbors = k)
scores = cross_val_score(knn, x_train,
y_train, cv = 10, scoring = 'accuracy');
MSE = [1-n for n in cv_scores)
  plt.figure()

best_k = k_list[MSE.index(max(MSE))]
print(best_k)
```

O/P:

optimal no. of neighbors = 3

Accuracy = 97.33%

Date 30/5

ANN using BP. LAB-7

```
import numpy as np
x = np.array ([[2,9], [1,5], [3,6]], dtype =
    float))
x = x/np.max (x, axis = 0)
  y = y/100
epoch = 5000
lr = 0.1
inputlayer_neurons = 2
hiddenlayer_neurons = 3
outputlayer_neurons = 1

wh = np.random.uniform (size = (inputlayer_
    neurons, hiddenlayer_neurons))
bh = np.random.uniform (size = (1, hidden
    layer_neurons))
wout = np.random.uniform (size = (hiddenlayer
_neurons, outputlayer_neurons))
bout = np.random.uniform (size = (1, outputlayer_neurons))
def sigmoid (x):
  return 1/(1+np.exp(-x))
def derivation_sigmoid (x):
    return x*(1-x)

for i in range (epoch):
hinpt = np.dot (x.wh)

hinp = hinp + bh
hlayer_ait = sigmoid (hinp)
outinpl = np.dot (hlayer_act.wout)
outinp = outinpl + bout
output = sigmoid (outinp)
```

```
hiddengrad = derivatives_sigmoid(hlayer_act)
d_hiddenlayer = Eh * hiddengrad
print("Input :\n" + str(n))
print("Actual output:\n", str(y))
print("Predicted output\n", output)
```

OUTPUT:

Input

```
[[0.66667  1 ]
 [0.333  0.55556
  1      0.66667 ]]
```

Actual output

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

Predicted output

```
[[0.7841   8718]
 [0.765    3553]
 [0.78892728]]
```

Random Forest Algorithm

```
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import iris
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_more
iris_data = pd.read_csv('Iris.csv')
iris_data.head()
iris_data.info()
iris_data.drop('Id', inplace=True, axis=1)
x = iris_data.drop('species', axis=1)
y = iris_data['species']
iris_data.info()

x_train, x_test, y_train, y_test = train_test_split(x,
                    y, test_size = 0.4, random_state=42)
rfc_classifier.fit(x_train, y_train)
```

Program 9b - AdaBoost

```
mylogregmodel = logistic Regression()
adabc = AdaBoostClassifier(n_estimator = 050,
                    base_estimator = mylogregmodel,
                    learning_rate = 1)
model = adabc.fit(x_test)
print('Accuracy score:', accuracy_score(y_test, y_pred))
```

O/P :

Accuracy = 1.0.

4/04/3015

Week-8

Program (10) - K means Algorithm to cluster a set of data stored in a .csv file.

```
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_length', 'petal_width']

y = pd.DataFrame (iris.target)
y.columns = ['Targets']
model = KMeans (n_clusters = 3))
model.fit (X)

plt.figure (figsize = (14,7))
plt.subplot (1,2,2)
plt.scatter [X['PetalLength cm'], X['PetalWidth cm'],
             C = model.labels, cmap = 'viridis',
             s = 40)

plt.show( )
```

(11) Lab - 11

```
cancer = load_breast_cancer()
cancer.keys()

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

scaler = StandardScaler()
scaler.fit(df)
scaled_data = scaler.transform(df)

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')
```

df.sf 30/5