Aim - 3.24

Date : 21/03
Page No :

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

Import pandas as pd

airbnb_data = pd.read_csv("data/listings_austin.csv")
airbnb_data.head()

Output :-
Imported the pandas library into our environment
Printed the first 5 rows of data frame

* Reading the data from URL

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_~~length~~ 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	petal...
0	5.1	3.5	1.4	0.2
1	4.9	3	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5	3.6	1.4	0.2

→ exporting dataframe to CSV file

iris_data.to_CSV ("cleaned_iris_data.csv")

Date: 21/03
Page No:

Kaggle

→ Creating data frame

```
Pd.DataFrame({'yes': [50, 21], 'No': [131, 2]})
```

output:

	yes	No
0	50	131
1	21	2

→ Series

```
Pd.Series([1, 2, 3, 4, 5])
```

0	1
1	2
2	3
3	4
4	5

dtype : int64

→ Reading Data Files

```
data = pd.read_csv("college - datacrest")

data = Pd.read_csv("college - data.csv")
data.head()
```

Returns first 5 data value of data set

Date: 28/3
Page No:

end to end project Hands on machine learning

Step 1:
get the data

data.head()
- gives first five data entries

data.info()
output:

	column	non-null
0	longitude	20640
1	latitude	20640
2	housing_age	20640
3	total_rooms	20640
4	total_bedrooms	20640
5	population	20640
6	households	20640
7	median_income	20640
8	median_house_income	20640
9	ocean_proximity	20640

data['ocean_proximity'].value_counts()
data.describe()

data.hist(bins=50, figsize=(20,15))
plt.show()

Date :
Page No :

Step 2:

Discover & visualize the data to gain insights

```
data.plot(kind='scatter', x='longitude', y='latitude')
plt.show
data.plot(kind='scatter', x='longitude', y='latitude',
          alpha=0.1)
plt.show
data.plot(kind='scatter', x='longitude',
          y='latitude', alpha=0.1)
plt.show()
data[['population', 'median_house_value']].corr()
corr_matrix = housing.corr()
corr_matrix = ['median_house_value'].sort_values
    (ascending = False)
data.plot(kind='scatter', x='median_income',
          y='median_house_value', figsize=(12,8)
          alpha=0.1)
plt.show()
```

Step 3: prepare the data for machine learning algo.

```
housing_num = data.drop("ocean_proximity",
                        axis=1)
```

housing_cat = housing[['ocean_proximity']]
housing_cat.head()
imputer.fit(housing_num)

for train_index, test_index in split(X=housing, y=housing['income_cat']):
strat_train_set = housing.loc[train_index]
strat_test_set = housing.loc[test_index]

housing = strat_train_set.copy()

5. Select and train model

from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(X=housing_prepared, y=housing_labels)
some_data = housing.iloc[:5]
print("Predictions:", lin_reg.predict(housing_prepared)

lin_mse = mean_squared_error(housing_labels, housing_predictions)

Date : 4/4
Page No :

Simple linear regression

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

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

Plt.scatter(df_sl['yrs experience'],
df_sl['salary'], color='right red')
plt.title('salary vs experience')
plt.show()

Split data

x = df_sl.iloc[:, :1]
y = df_sl.iloc[:, 1:]

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)
```

y_pred_test = regressor.predict(x_test)
y_pred_train = regressor.predict(x_train)

Visualizations

plt.scatter(x_train, y_train, color='lightcoral')
plt.plot(x_train, y_pred_train)
plt.show

Coefficient and intercept

print('Coefficient: ', regressor.coef_)
print('Intercept: ', regressor.intercept_)

coefficient: [[9312.57512]]
intercept: [26780.099]

Date : 4/4
Page No :

Multiple linear regression

Import pandas as pd
Import numpy as np
Import seaborn as sns

from sklearn.linear_model import LinearRegression

df_star = pd.read_csv('Star_ups.csv')
df_star.head()
df.describe()

distribution

plt.title('profit distribution plot')
sns.distplot(df_star['profit'])
plt.show()

Relationship between profit and R&D spend

plt.scatter(df_star['R&D Spend'], df_star['profit'])
plt.show()

Split data

X = df_star[:, :-1].values
y = df_star[:, -1].values

Split into train data

Data :
Page No :

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)

Date: 18/4
Page No:

Decision tree using entropy

```
Import pandas as Pd
iris_data = Pd.read_CSV('Iris.CSV')
Iris_data.head()

Import pandas as Pd
from Sklearn.model_Selection import train_test_split
from Sklearn.tree import decisionTreeClassifier
from Sklearn.metrics import accuracy_score
from Sklearn.tree import decisiontreeclassifier, plot_tree
import matplotlib.pyplot as Plt

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=0.6, Random_State=42)

dt_Classifier = decision tree classifier (criterion = 'entropy',
        random_state = 42)

dt_Classifier.fit(X_train, y_train)
y_pred = dt_Classifie.predict(X_test)

accuracy = accuracy_score(y_test, y_Pred)
print(f'accuracy : {accuracy}')
```

Data :
Page No :

petal length cm <= 2.45
entropy = 1.581
Samples = 96
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]
Class = iris-virginica

Date : 25/4
Page No :

Build logistic regression model for a given dataset

```
import pandas as pd
from matplotlib import pyplot as plt
%matplotlib inline
df = pd.read_csv("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=0.8)
```

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

y_predicted = model.predict(X_test)

print(y_predicted)
[1 1 0 1 0]

print(X_test)

    age
9    61
23   45
21   26
```

4	46
7	80
6	22

Print ("Model Accuracy", model.score (x_test, y_test))

model Accuracy 0.833

print ("coefficient", model.coef_)
print ("intercept", model.intercept_)

Coefficient [[0.1499]]
intercept [[-5.6631]]

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)

0.485004

age = 43
prediction_function (age)
0.568565

Date: 9/05
Page No:

SVM & KNN

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

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

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

plt.scatter(iris_data['SepalLengthCm'], iris_data['SepalWidthCm'],
    c = iris_data['Species'], cmap = 'viridis')
plt.xlabel('length (cm)')
plt.ylabel('width (cm)')
plt.title('Scatter plot of iris data set')
plt.colorbar(label='Species')
plt.show()

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 = 0.3, random_state = 42)
```

Date :
Page No :

```
svm_classifier = SVC()
svm_classifier.fit(X_train, y_train)

y_pred = svm_classifier.predict(X_test)

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

Accuracy : 0.9333

for KNN

Accuracy : 0.9833

3. Ada boost

```
from sklearn.linear_model import logistic regression
from sklearn.ensemble import adaboostclassifier
from sklearn import metrics
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split

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_data_split
        (X, y, test_size = 0.4, random_state = 42)

mylogreg model = logistic regression()
adabc = adaboostclassifier(n_estimators = 156, base_
        = mylogreg model, learning_rate=1)

model = adabc.fit(X_train, y_train)
y_pred = model.predict(X_test)
metrics.accuracy_score(y_test, y_pred)
```

0.9633

Date: 23/05
Page No:

Random forest

Pip install scikit-learn

```
Import pandas as pd
from Sklearn.model_Selection import train_test_split
from Sklearn.ensemble import RandomForestClassifier
from Sklearn.datasets import load_iris
from sklearn.metrics import accuracy_score

Iris_data = pd.read_csv('Iris.csv')
Iris = data.load()
Iris = data.info()

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)

rf_classifier = RandomForestClassifier(n_estimators=100,
                random_state = 42)

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

Date: 23/05
Page No:

Dec-15-24

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

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,
                               neurons))
bout = np.random.uniform(size=(1, output_neurons))

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

```
outinp = outinp1 + bout
output = Sigmoid(outinp)

EO = y - output
outgrad = derivatives_Sigmoid(output)
d_output = EO * outgrad

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: " + str(X))
print("Actual output: " + str(y))
print("Predicted output: ", output)
```

30-5-24

Date : 30/05
Page No :

K-means 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_length', 'petal_width']
y = pd.DataFrame(iris.target)
y.columns = ['Targets']

model = kmeans(n_clusters=3)
model.fit()

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 lengths')
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')
```

PCA

```
Import matplotlib.pyplot as plt
Import pandas as pd
Import numpy as np
Import Seaborn as sns

from sklearn.datasets import load_breast_cancer
cancer = load_breast_cancer()
cancer.keys()
Print (cancer['DESCR'])

df = pd.DataFrame(cancer['data'], columns = cancer[
                                              - na
df.head()
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
(569, 2)
```

Import matplotlib.pyplot as plt

plt.figure(figsize=(8,6))
plt.scatter(x_pca[:,0], x_pca[:,1], c=cancer['target'], cmap='plasma')
plt.xlable('first principle component')
plt.ylable('Second principle component')
plt.title('PCA of breast cancer data set')
plt.colorbar(lable='cancer type')
plt.show()