Teacher Sign / Remarks

Ex-3-24 Date / /

Page

21/03/2024

Lab-1

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

i) Download listings-austin.csv file.

Code:

```
import pandas as pd
airbnb-data = pd.read_csv("C:/Users/Admin/Downloads/listings-austin.csv")

airbnb-data.head()
```

#output:

	id	listing_url	scrape_id	last_scraped
0	2265	https://www.airbnb.com/rooms/2265	20180710171409	2018-07-10
1	5245	https://www.airbnb.com/rooms/5245	20180710171409	2018-07-10
2	5456	https://www.airbnb.com/rooms/5456	20180710171409	2018-07-10
3	5769	https://www.airbnb.com/rooms/5769	20180710171409	2018-07-10
4	6413	https://www.airbnb.com/rooms/6413	20180710171409	2018-07-10

5 rows x 96 columns

21/03/2024

Date __/__/__
Page

2) Reading Data from URL:

Code:

```
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()
```

#Output:

	class	sepal_length_in_cm	sepal_length_in_cm	petal_length_in_cm	petal length
0	iris_setosa	5.1	3.5	1.4	0.2
1	iris_setosa	4.9	3.0	1.4	0.2
2	iris_setosa	4.7	3.2	1.3	0.2
4	iris_setosa	4.6	3.1	1.5	0.2
5	iris_setosa	5.0	3.6	1.4	0.2

→ Export code:

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

Date __/__/__

Page 150/25

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	It was awful
Product B	Pretty good	Bland

Reading Data Files.

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

2) wine_reviews.shape
(129971, 14)

3) wine_reviews.head

Date __/__/__
Page

28/03/24.

End to End hands on machine learning:

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()
```

Step 2: Discover and visualize the data to gain sight

```
data.plot(kind='scatter', x='longitude',
          y='latitude')
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 algorithms.

```
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.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()
```

Date __/__/__
Page

5. Select and train model:

```
from sklear.linear_model import LinearRegres
lin_reg = LinearRegression()
lin_reg.fit(X= housing_prepared, y= housing
                                    -labels)

some_data = housing.iloc[:5]
print("Predictions:", lin_reg.predict(some_data)

from sklearn.metrics import mean_squared_er
housing_prediction = lin_reg.predict(housing_
                                        prepared)
lin_mse = mean_squared_error(housing_labels
                             housing_predictio
```

6. Fine-Tune your model.

```
from sklearn import RandomForestRegressor
from sklear.model_selection import GridSearch
forest_reg = RandomForestRegressor()
grid_search.fit(X= housing_prepared,
                Y= housing_labels)

grid_search.best_params
grid_search.best_estimator
```

4/04/2024

Week-3

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_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['Years Experience'],
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)
```

Date
Page

Train model

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

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_}')
```

Output:

Coefficient: [[9312.57512]]
Intercept: [26780.09911]

Date / /

Page

Multiple 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_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 and RandD spend

plt.scatter(df_start['RandD Spend'], df_start['Profit'])
plt.show()

Spit Data
X = df_start[:, :-1].values
y = df_start[:, -1].values

# split into train test data.
X_train, y_train, X_test, 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 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, 103282.38],
       [132552.28, 144259.4],
       [132447.74, 146121.95],
       ])
```

18/04/2024

Week-4

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 sklear.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)
```

```
# 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.da
          feature_names, class_names=iris.data.
                                     target.na
plt.show()
```

Output:

Accuracy 1.0

25/04/2024

Week-5

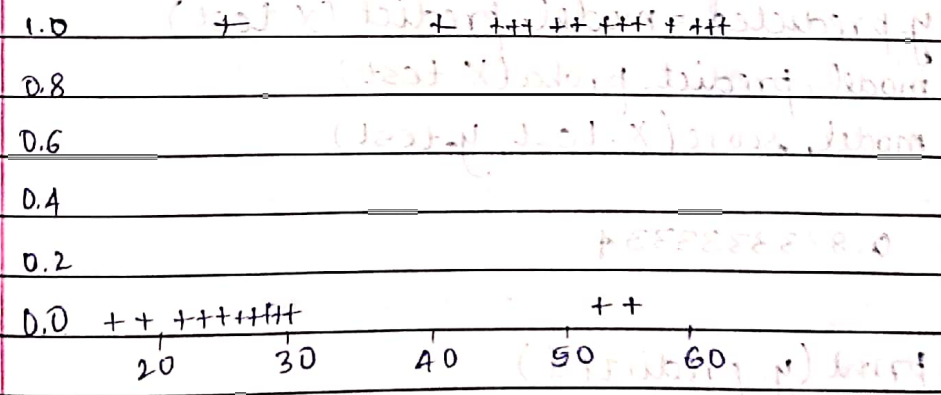
Build logistic regression model for given dataset

Code:

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

	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')
from sklearn.model_selection import train_test_split
```



P.T.O.

```
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)
```

- LogisticRegression

```
LogisticRegression()

print(X_test)
```

	age
26	23
25	54
11	28
1	25
16	25
18	19

```
y_predicted = model.predict(X_test)
model.predict_proba(X_test)
model.score(X_test, y_test)
```

0.8333333334

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

9/05/2024 Dg-5-24

Date __/__/__

Page _____

Lab program no 6,
Build KNN classification model for given dataset.

Python code

```
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

label_mapping

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

	Id	SepalLength	SepalWidth	PetalLength	PetalWidth	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

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

P.T.O.

```
plt.scatter(iris_data['sepallength 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 Feature
plt.colorbar(label='species')
plt.show()
```

```
iris_data['species'].unique()
```

Output:

```
array([1, 2, 3], dtype=int64)
```

```
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)
```

```
knn = KNeighborsClassifiers(n_neighbors = 5)
knn.fit(X_train, y_train)
```

▽ KNeighborsClassifier

KNeighborsClassifier

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

Output:

Accuracy: 0.9833333

23/05/2024 A 23-5-24

Week no-7

Implementation of ANN using back Propagation for given values

Code:

```
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  # training iterations.
  lr = 0.1  #. learning rate
  input_layer_neurons = 2
  hidden_layer_neurons = 3
  output_neurons = 1

# weight and bias initialization

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))

# sigmoid function
def sigmoid(x):
    return 1/(1+np.exp(-x))
# Derivative
    of sigmoid function
```

```
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, wout)
    outinp = outinp1 + bout
    output = sigmoid(outinp)

    EO = y - output
    outgrad = derivatives_sigmoid(output)
    d_output = EO * outgrad
    EH = d_output.dot(wout.T)
    hiddengrad = derivatives_sigmoid(hlayer_ac
    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)
```

Output:

```
Input:
[[0.66666 7    1.
  0.333333     0.555556
  1.           0.666667]]
```

Date __/__/__

Page _____

Actual Output:

[[0.92]
 [0.86]
 [0.89]]

Predicted output

[[0.88555471],
 [0.88108002],
 [0.88579557]]

Date / /
Page

23/05/24

Program - 9a

```
from sklear.ensemble import RandomForestClassifi
import pandas as pd
from sklearn.metrics import accuracy_score

df = pd.read_csv("melb_data.csv")
melbourne_data = df.dropna(axis=0)
y = melbourne_data.Price
melbourne_features = ['Rooms', 'Bathroom',
                      'Landsize', 'BuildingArea',
                      'YearBuilt', 'Latitude',
                      'Longitude']
X = melbourne_data[melbourne_features]

from sklearn.model_selection import train_test_sp
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)
```

Output:

RandomForestClassifier()

```
y_predicted = model.predict(X_test)
accuracy = accuracy_score(y_predicted, y_test
print(accuracy)
```

Output:

0.9935483870967742

A 30/5/24

30/05/2024

K means Clustering Algorithm

implementation code:

```
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)
```

Output

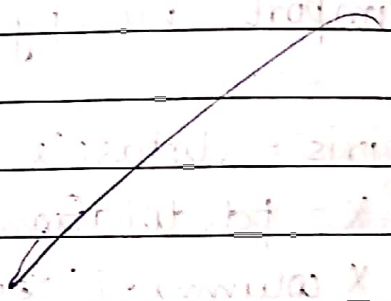
```
KMeans
KMeans(n_clusters=3)
```

```
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('K Means Clustering')
plt.xlabel('Petal_Length')
plt.ylabel('Petal_Width')
```

Date / /
Page

```
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')
```



Date __/__/__

Page _____

30-05-2024

PCA Visualization:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(df)
```

Output:

```
StandardScaler(copy=True, with_mean=True,
               with_std=True)
```

```
Scaled_data = scaler.transform(df)
```

```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
pca.fit(Scaled_data)
```

Output:

```
PCA(copy=True, n_components=2, whiten=False)
```

```
x_pca = pca.transform(Scaled_data)
```

```
Scaled_data.shape
    (569, 30)
```

```
x_pca.shape
    (569, 2)
```

```
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')
```