Date 21/03/2024

Saathi

Write a python program to import and export data using Pandas lib. fun's

```
import pandas as pd.
airbnb_data = pd.read_csv("listings-austin.csv")
airbnb_data.head()
# Imported Data

# Reading 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-width-in-cm", "class"]
iris_data = pd.read_csv(url, names=col_names)
iris_data.head()

# Exporting DF to a CSV file
iris_data.to_csv("cleaned-iris_data.csv")
```

Output:

	id	listing_url	scrape_id	....	cal_host_listings_count
0					
1					

	sepal_length_in_cm	width	petal_length_in_cm	width	class
0					
1					

21/03/24

Saathi

Date 28/03/24

I 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 data. →

```
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()
```

Creating a Test Set

→

```
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), len(test_set)
```

Page No.

Date ___/___/_____

Saathi

→ 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])

Visualizing
housing['income_cat'].hist()
strat_test_set['income_cat'].value_counts()/len(strat_test_set)

for set_ in (strat_train_set, strat_test_set):
    set_.drop('income_cat', axis=1, inplace=True)

→ # We tried to generate test set ∵ it is an imp part of any ML project.

II Discover & Visualize the Data to Gain Insights

→ housing = strat_train_set.copy();
housing.shape
housing.plot(kind='scatter', x='longitude', y='latitude')
plt.show()
housing.plot(kind='scatter', x='longitude', y='latitude', alpha=0.1)
plt.show()
housing.plot(kind='scatter', x='longitude', y='latitude', alpha=.4, s=housing['population']/100., label='population', figsize=(10,7), c='median_house_value', cmap=plt.get_cmap('jet'), colorbar=True)
plt.legend()

Page No.

Date __/__/____

Saathi

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

→ Correlations:

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()
- housing.plot(kind = 'scatter', x = 'median_income',
y = 'median_house_value', figsize = (12,8),
alpha = 0.1)
plt.show()

→ corr_matrix = housing.corr()
corr_matrix['median_house_value'].sort_values
(ascending = False)

III Preparing The Data for ML Algorithms
1. Data cleaning
IV 2. Handling Text and Categorical Attributes
3. Custom transformers
4. Feature Scaling
5. Transformation Pipelines

Page No.

Date __/__/____

Saathi

IV Select and Train a Model.

1. Training & Evaluating on the Training set.
2. Better Evaluation using Cross-Validation

V Fine-Tune Your model.

1. Grid Search
2. Randomized Search
3. Ensemble Methods.
4. Analyze models.
5. Evaluating System

VI Exercises

28/3/24

Date 4/4/24

Saathi

Lab Program No. 4

Simple Linear Regression

```
df_sal = pd.read_csv('Salary_Data.csv')
df_sal.head()

df_sal.describe

plt.title('Salary Dist plot')
sns.distplot(df_sal['Salary'])
plt.show()

plt.scatter(df_sal['Years Exp'], df_sal['sal'],
color = 'lightcoral')
plt.title('sal v/s Exp')
plt.xlabel('Years of Exp')
plt.ylabel('Salary')
plt.box(False)
plt.show()

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

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

regresson = LinearRegression()
regresson.fit(X_train, y_train)

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

Date __/__/____

Saathi

```
plt.scatter(X_train, y_train, color='light
                                    coral')
plt.plot(X_train, y_pred_train)
plt.show()

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

Page No.

Saathi

Date __/__/____

Multiple Linear Regression

```
df_start = pd.read_csv('50_Startups.csv')
df_start.head()

df_start.describe()

plt.title('Profile Dist Plot')
sns.distplot(df_start['Profit'])
plt.show()

plt.scatter(df_start['R&D Spend'],
            df_start['Profit'], color = 'lightcoral')
plt.box(False)
plt.show()

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

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 = regressor.predict(X_test)

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

Done 7/12/24

Page No. ____

Date __/__/____

Saathi

Building the decision tree (IO3) and apply 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 = 'gini')

clf.fit(X_train, y_test)

y_pred = clf.predict(X_test)

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

plt.figure(figsize = (12,8))
plot_tree(clf, feature_names = feature_names)
          class_names = target_names, filled = True)
plt.show()
```

Output: Accuracy = 0.983

Page No.

Date 25/04/24

Saathi

Lab Program no.5

Build Logistic Regression Model for a given data

```
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=.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)
```

→ Model accuracy : 0.8333333334

Analyzing through Linear regression

```
model.coef_
```

→ Coefficient (slope) : [[0.14544708]]

```
model.intercept_
```

→ Intercept : [-5.44733781]

Date __/__/____

Saathi

Analyze through sigmoid function.

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

def prediction_function(age):
    z = 0.042 * age - 1.53 # 0.04150133 ~
        0.042 and -1.52726963 ~ -1.53
    y = sigmoid(z)
    return y

age = 35
prediction_function(age)
```

→ Prob: 0.485004498380 05899

```
age = 43
prediction_function(age)
```

→ Prob: 0.5685652990777205

25/4/24

Saathi

Date 09/05/24

Lab program No. 6

Build KNN classification model for a given dataset.

```
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.show()

iris_data['Species'].unique()

array(['Iris-setosa', 'Iris-versicolor',
       'Iris-virginica'], dtype=object)

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 = KNeighboorsClassifier(n_neighbors=5)
Knn.fit(X_train, y_train)
```

```
y_pred = knn.predict(X_test)

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

- Accuracy : 0.9833

Ay 9/5/24

Date 9/5/24

Saathi

SVM model

```
svm_classifier = SVC()
svm_classifier.fit(X_train, y_train)
y_pred = svm_classifier.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
```

Accuracy : 0.9333

Done 9/5/24

Date 23/5/24

Saathi

ANN program no. 8

```
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

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

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

for i in range (epoch):
    #Forwd  hinp1 = np.dot (X, wh)
    #propagation.  hinp = hinp1 + 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
```

Saathi

Date __/__/____

updating weights and biases.

wout += hlayer_act.T.dot(d_output) * lr.

wh += X.T.dot(d_hiddenlayer) * lr.

print("Predicted o/p: \n", output)

predicted o/p:

[[0.89]
[0.87]
[0.89].].

Date __/__/____

Saathi

Random Forest Algo Lab 9a&b

```
data = pd.read_csv ('Iris.csv')
X = data.drop (columns = ['Species'])
y = data ['Species'].

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

rf_classifier = RandomForest (Classifier
    (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 * 100:.2f}%')
```

Accuracy: 100.00%.

23/5/24

Date __/__/____

Saathi

Week 8. Lab program -10

```
iris =
data = pd.read_csv('Iris.csv')
X = data[['SepalLengthCm', 'SepalWidthCm',
          'PetalLengthCm', 'PetalWidthCm']]
model = KMeans(n_clusters = 3))
model.fit(X)

plt.figure(figsize = (14,7))

plt.subplot(1,2,2)
plt.scatter(X['PetalLengthCm'], X['PetalWidthCm'],
            c = model.labels_, cmap = 'viridis',
            s = 40)
plt.show()
```

Page No.

Date __/__/____

Saathi

Week 8 Lab program -11

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

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

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')
```

30/5/24