INDEX

Sl. no		Title		Sign
21/3/24	1.	Import & export data using Pandas Library	-10	
28/3/24	2.	Various Data preprocessing techniques	-10	
4/4/24	3.	Simple & multiple linear regression	-10	
18/4/24	4.	ID3	-10	
25/4/24	5.	Logistic regression	-10	
9/5/24	6.	KNN & SVM	-10	
23/5/24	7.	ANN, ~~RandomForest~~		
23/5/24	8.	RandomForest, AdaBoost	-10	
30/5/24	9.	Clustring kmeans	-10	
30/5/24	10.	Dim Red → PCA	-10	

21/3/24 LAB 1

TO import & export data using Pandas library functions

(i) importing data

```
import pandas as pd
airbnb_data = pd.read_csv("data/listings-austin.csv")
airbnb_data.head()
```

(ii) 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()
```

(iii) Exporting dataframe to csv file

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

OUTPUT:

	sepal-length-in-cm	sepal-wid-in-cm	petal-length-in-cm	petal-wid-in-cm	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1					
2					
3					
4					

21/3/24

28/3/24 End to End Machine Learning Project-

1. Look at the Big picture
2. Get the Data

(i) Download the data

```
import os
import tarfile
import urllib
DOWNLOAD_ROOT = "https://raw.github.com"
housing_path = os.path.join("data", "01")
housing_url = download_root + "datasets/housing/housing.tgz"
def fetch_housing(housing_url = housing_url,
                  housing_path = housing_path):
    os.makedirs(name = housing_path, exist_ok =
                                        True)
    tgz_path = os.path.join(housing_path,
                            "housing.tgz")
```

fetch_housing_data()

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

```
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 & Visualize the Data to gain insights

```
strat_train_set.shape, strat_test_set.shape
strat_test_set.reset_index().to_feather(frame =
                                'data/01/strat_test_
                                set.ft')
housing = strat_train_set.copy();
housing.shape
housing.plot(kind='scatter', x='longitude', y='latitude')
plt.show()

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

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
X = imputer.transform(housing_num)
X.shape

housing_tr = pd.DataFrame(data=X, index=housing_num
                          columns=housing_num.columns)
housing_tr.head()

housing_cat = housing[['ocean_proximity']]
housing_cat.head()

from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler

num_pipeline = Pipeline([
    ('imputer', SimpleImputer(strategy='median')),
    ('attribs_adder', CombinedAttributesAdder()),
    ('std_scaler', StandardScaler()),
])

housing_num_tr = num_pipeline.fit_transform
                 (housing_num)

housing_num_tr.shape
```

28-3-24

5. select & 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_labels, 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
```

7. Launch, monitor & maintain your system.

4/4/24

Week 3

SLR

```
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
 color = 'lightcoral'
plt.title('sal v/s Exp')
plt.ylabel('Salary')
plt.xlabel('Years of Exp')
 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)
regression = LinearRegression()
regression.fit(X_train, y_train)
plt.scatter(X_train, y_train, color = 'lightc')
plt.plot(X_train, y_pred_train)
print(f'coeff: {regression.coef_}')
print(f'Intercept: {regressor.intercept_}')
```

Multiple Linear Regression

```
df_start = pd.read_csv('50_Startups.csv')
df_start.head()
df.start.describe()
plt.title('Profile Dist Plot')
plt.show

plt.scatter(df_start['R&D Spend']
df_start['Profit'].colour = 'light coral')
plt.box(false)
plt.show()

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

regression = LinearRegression()
regressor.fit(X_train, y_train)
y_pred = regressor.predict(X_test)

result = np.concatenate((y_pred.reshape
(len(y_pred), 1), y_test.reshape(len(ytest),1)
print(result)
```

18/11/24

Week 4

ID3

```
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').
accuracy = clf.score(X_test, y_test)
print("Accuracy", accuracy)
plt.figure(figsize = (12,8))
plt.show()
```

O/P:

Accuracy = 0.983

25/4/24 Wednesday

Logistic regression

```
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
# Perform the split
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)
y_predicted = model.predict(X_test)
model.predict_proba(X_test)
model.score(X_test, y_test)
```

Output:

Accuracy: 0.8333333333333334

Analyse through Linear Regression

model.coef_ - indicates value of m in y = mx + b eq.

model.coef_ - coef (slope) = [[0.1454477 08]]

model.intercept_ - indicates value of b in y = mx + b

model.intercept_ : [-5.44733781]

Analyse through sigmoid function:

import math

def sigmoid(x):

return 1/(1 + math.exp(-x))

def prediction_function(age):

z = 0.042 * age - 1.53 # 0.041501 33 - 0.042 and -1.5272696 3 ~ -1.53

y = sigmoid(z)

return y

age = 35

prediction_function(age)

prob: 0.485004 498 38 058 99

age = 43

prediction_function(age)

prob: 0.568 565 2990 7770 5

25-4-2022

Week6

7/5/24

KNN Classifier:

① import numpy as np
import pandas as pd
import matplotlib as plt
plt.style.use('ggplot')

② df = pd.read_csv('/diabetes.csv')
df.head
df.shape

③ X = df.drop('outcome', axis=1).values
y = df['outcome'].values

④ from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test =
train_test_split(X, y, test_size=0.4,
random_state=42,
stratify=y)

⑤ from sklearn.neighbors import KNeighboursClassifier
neighbors = np.arange(1,9)
train_accuracy = np.empty(len(neighbors))
test_accuracy = np.empty(len(neighbors))
knn.fit(X_train, y_train)

train_accuracy[i] = knn.score(X_train, y_train)
test_accuracy[i] = knn.score(X_test, y_test)

(6) plt.plot(neighbors, train_accuracy, label='Train')
plt.plot(neighbors, test_accuracy, label='test')
plt.show()

(7) plt.scatter(neighbors, train_accuracy, color='k')
plt.show()

(8) knn.fit(X_train, y_train)
knn.score(X_test, y_test)

O/P Accuracy = 0.7348

Lec
9-5-24

23/5

SVM

```
from sklearn import datasets
cancer = datasets.load_breast_cancer()
# cancer features & print
print(cancer.target)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test =
train_test_split(cancer.data, cancer.target,
    test_size = 0.3, random_state = 109)
from
import sklearn import svm
clf = svm.SVC(kernel = 'linear')
y_pred = clf.predict(X_test)

from sklearn import metrics
print(metrics.accuracy_score(y_test, y_pred))
print(metrics.precision_score(y_test, y_pred))
print(metrics.recall_score(y_test, y_pred))

import seaborn as sns
sns.scatterplot(x = cancer.data[:, 0]
    y = cancer.data[:, 1], hue = cancer.target)
plt.show()
```

O/P Accuracy : 0.9764

4.5m

28/8/24 Week 7

Artificial neural network (single)

```
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
outputlayer_neurons = 1
wh = np.random.uniform(size=(inputlayer_neurons, hiddenlayer_neurons))
bh = np.random.uniform(size=(1, hiddenlayer_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)
    outinp = sigmoid(outinp)
    EO = y - output
    outgrad = derivatives_sigmoid(output)
    EH = d_output.dot(wout.T)
```

hiddengrad = derivatives - sigmoid (Hayer-act)

d-hiddenlayer = Err * hiddengrad

wh += X.T . dot (d-hiddenlayer) * lr

// print I/p & O/p

O/p :

Input-

[[0.66667 1.]
[0.3333 0.5556]
[1. 0.66667]]

Actual Output:

[[0.92]
[0.86]
[0.89]]

Predicted Output-

[[0.81969177]
[0.79692237]
[0.81616067]]

Iva
23-5-24

Prog 7

9a) Random forest -

```
// import libraries
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 = 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 accuracy, classification report

feature_importances = rf_classifier.feature_importances_
features = X.columns
importance_df = pd.DataFrame({'Feature': features,
                              'Importance': feature_importance}).
                              sort_values(by='Importance',
                                          ascending=False)
print(importance_df)
```

Op: Accuracy : 100.00%.

23-5-24

9b) AdaBoost

```
// import necessary libraries
data = pd.read_csv('/content/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)
adaboost_clf = AdaBoostClassifier
        (n_estimators = 150, learning_rate = 1)
adaboost_clf.fit(X_train, y_train)
y_pred = adaboost_clf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
// print Acc
```

Op:

Accuracy : 1.0

23-5-24

30/8/24 Week 8

LAB PROG 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, 14))
colormap = np.array(['red', 'lime', 'black'])
plt.subplot(2, 2, 1)
plt.scatter(X.Petal_Length, X.Petal_Width
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
plt.title('K-Means Clustering')
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
```

23/5/24

Week 8

LAB 11

Dimensionality reduction using PCA

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

from sklearn.datasets import load_breast_cancer
cancer = load_breast_cancer()
cancer.keys()
print(cancer['DESCR'])
df = DataFrame(cancer['data'], columns = cancer['feature_names'])
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)
PCA()
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 PC')
```

25/5/24