INDEX

NAME: Tanisha. Gotadke STD: 6th sem SEC: D ROLL NO: 1BM21CS229

S. No.	Date	Title	Page No.	Teacher's Sign / Remarks
01	21/03	Week 1	01	21/3/24
02	28/03	Week 2	02	4/4/24
03	04/04	Week 3	05	4/4/24
04	18/04	Week 4	08	18/4/24
05	25/04	Week 5	10	25/4/24
06	09/05	Week 6	13	23/5
07	23/05	Week 7	15	30/5
08.	30/05	Week 8	20	30/5

21/03/2024

Week-1

Import and Export a csv file with pandas in python

Code:

```
import pandas as pd
df=pd.import_csv("file-path.csv")
df.head()
```

Output:

Id	"Sepal length	Sepal Width	Petal length	class
0	5.1	3.5	1.4	iris-sefora
1	4.9	3.0	1.4	iris-sefora
2	4.7	3-2	1.3	iris-sefora
3	4.6	3.1	1.5	iris-sefora
4	5.0	3.1	1.4	iris-sefora
5	3.9.	3.6	1.2	iris-sefora.

Export Code:

```
df.to-csv("path\new-name-csv")
```

Reading data from URL.

```
# Define the col names
col_names=["sepal-length-in-cms", "sepal-width-in-cms", "petal-length-in-cms", "petal-width-in-cms", "class"]
# Read data from URL
iris-data=pd.read-csv(url, names=col-names)
iris_data.head()
```

21/3/24

Week 2

1. Performance Measure
2. Get the Data
2.1. Download the data.

```
import os
import tarfile
import urllib

DOWNLOAD_ROOT = "https://raw.githubusercontent.com/ageron/handson-ml2/master/"
HOUSING_PATH = os.path.join("datasets", "housing")
HOUSING_URL = DOWNLOAD_ROOT + "datasets/housing/housing.tgz"

def fetch_housing_data(housing_url=HOUSING_URL, housing_path=HOUSING_PATH):
    os.makedirs(name=housing_path, exist_ok=True)

# Download 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)
```

Create Test Set:

Train - 80%.

Test - 20%

```
# Using Numpy
import numpy as np
```

```
def split_train_test(data, test_ratio=0.2):
    shuffled_indices = np.random.permutation(len(data))
```

Step 2: Discover and Visualize the Data

```
train_set.shape, test_set_train.shape

#Scatter plot
housing.plot[kind='scatter', x='longitude',
y='latitude') plt.show()

#correlation
housing[["population", "median_house_value"]].corr()
```

Step 3: Prepare the data for Machine Learning Algorithms.

```
housing = strat_train.drop("median_house", axis=1)
housing_sobers = strat_train_set("median_value".
                                 copy()
housing_shape, housing_robus.shape

#Data Cleaning
housing.dropna(subset='total_bedrooms')
```

Step 4: Select and train Model

```
from sklearn.linear_model
    import LinearRegression
    lin_reg = LinearRegression()

from sklearn.metrics import mean_squared_error
housing_predictions = lin_reg.predict(housing_prepared)
```

Better Evaluation

```
from sklearn.model_selection import cross_val_score
```

Step 5: Fine Tune Model

from sklearn.model_selection import GridSearchCV

final_model = grid_search.best_estimator_

final_predictions = final_model.predict(X = X_test_Prepared).

04/04/24

Date
Page 05

Week - 3

Implement linear and Multiple linear Regression algorithm using appropriate dataset.

Linear Regression.

(1)
```
df-val = pd.read_csv(r'c:\users\STUDENT...csv')
df_val = head()
```

(2)
```
plt.title('Salary-Distributism Plot')
sns.distplot(df-val['salary'])
plt.show()
```

(3)
```
plt.scatter(df-sal['Years Experience'],
df-sal['salary'], color='lightcoral')

plt.xlabel('Years of Exp')
plt.ylabel('salary')
plt.box(False)
plt.show()
```

(4) Splitting variables
```
x=df-sal.iloc[:,:]
y=df-sal.iloc[:,1:]
```

(5)
```
x-train, x-test, y-train, y-test = train_test_
split(x, y, test_size=0.2, random_state=0)
```

(6)
```
Regressor.get(x-train, y-train)
```

(7) Prelim

5

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. df_start = pd.read_csv("/content/50_Startups.csv")
   df_start.head()

2. plt.title('Profit Distribution Plot')
   sns.displot(df_start['Profit'])
   plt.show()

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

4. x_train, x_test, y_train, y_test = Train_Train_split(x, y, test_size=0.2, random_state=0)

5. Regressor = LinearRegression()
   regressor.fit(x_train, y_train)

6. Predict
   y_pred = regressor.predict(x_test)

Date
Page 07

Steps:

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

07/11/24

18/04/2024

Week-4

```
import pandas as pd
from sklearn.model_selection import train_test_spl
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import DecisionTreeClassifier,
plot_tree
from sklearn.metrics 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)
```

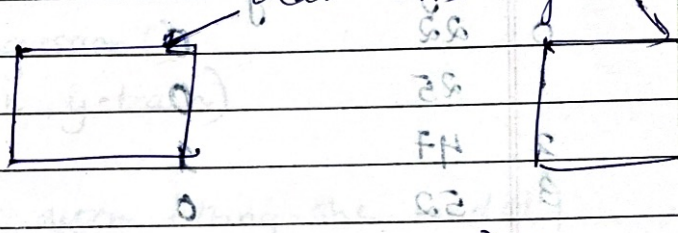
DecisionTreeClassifier
DecisionTreeClassifier (criterion = 'entropy',
random_state = 42)

```
plt.figure(figsize=(12,8))
plot_tree(dt_classifier, feature_names=X.columns, class_names=dt_classifier.classes_, filled=True)
plt.show()
```

petalLength(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

petalWidth(cm) <= 1.75
entropy = 1.0
samples = 63
value = [0, 31, 32]
class = Iris-virginica



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

Accuracy: 0.9833

18/4/24

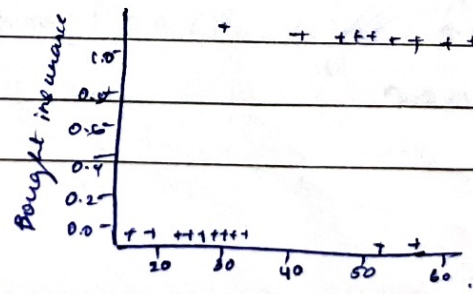
25/04/2024

Week - 5

```
import pandas as pd
from matplotlib import pyplot as plt
from sklearn.model_selection import train_test
_split
from sklearn.linear_model import Logistic
Regression

#Read the data:
df = pd.read_csv("insurance_data.csv")

#Display the first few rows of the Df
print(df.head())
```

	age	bought_insurance
0	22	0
1	25	0
2	47	1
3	52	0
4	46	1

```
#Scatter plot
plt.scatter(df.age, df.bought_insurance,
marker='+', color='red')
plt.xlabel('Age')
plt.ylabel('Bought Insurance')
plt.title('Age vs Bought Insurance')
plt.show()
```



```
X_train, X_test, y_train, y_test =
train_test_split(df[['age']], df.bought_insurance,
train_size=0.8)

print("X_test:")
print(X_test)
```

X_test:

	age
14	49
24	50
16	25
2	47
6	55
22	40

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

print("\n X_test after fitting the model:")
print(X_test)

y_predicted = model.predict(X_test)
print("\n Predicted values:")
print(y_predicted)
```

X_test after fitting the model:

	age
14	49
24	50
16	25
2	47
6	55
22	40

Predicted values:

[1 1 0 1 1 0]

Probabilities
print ("\n Probabilities:")
print (model.predict_proba (X_test))

Model Accuracy
print ("\n Model Accuracy:")
print (model.score (X_test, y_test))

Probabilities:
[[0.21294835 0.78705165]
[0.17694536 0.82305464]
[0.98538887 0.01461113]]

Model Accuracy:
0.5

25/4/24

09/05/2024

Date
Page 13

Week 6

KNN and SVM

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

Read the Dataset
df = pd.read_csv ("/content/Iris.csv")

Show the Plot:
plt.show()
plt.figure (figsize = (10, 6))
sns.Scatterplot (data = df, x = df ["Petal Length"]
                 y = df ["Sepal Width"]
                 legend = False, Hue = df ["Species"])

Add Labels and Title to Plot:

plt.xlabel ("Petal Length")
plt.ylabel ("Sepal Width")
plt.title ("Petal Length vls Sepal Width")
plt.show()

KNN:
from sklearn.model_selection import test_train_
split as tts,
from sklearn.neighbours import KNeighbourClassifier
X_train, X_test, y_train, y_test = tts (x, y, 0.6, 0)
neigh = KNeighbourClassifier (n_neighbours = 2)
neigh.fit (x, y)
```

```
y_pred = neigh.predict(x_test)
acc = accuracy_score(y_pred, y_test)
print(acc)
→ 1.0
```

SVM

```
from sklearn.svm import SVC
model = SVC()
model.fit(x, y)
y_pred = model.predict(x_test)
acc = accuracy_score(y_pred, y_test)
print(acc)
1.0
```

23/5/24

Date
Page 15

23/05/2024

Week-7

Random Forest

pip install scikit-learn

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

iris = load_iris()
x = iris.data
y = iris.target

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.5, random_state=32)

rf_classifier = RandomForestClassifier()
rf_classifier.fit(x_train, y_train)

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

Accuracy = 0.93333

```
classification_rep = classification_report(y_test, y_pred)

print("\n Classification Report \n",
      classification_rep)
```

Output

6

Output:

Classification Report

	Precision	Recall	f1-score	Support
0	1.00	1.00	1.00	30
1	0.81	1.00	0.89	21
2	1.00	0.79	0.88	24
Accuracy			0.93	75
Macro avg	0.94	0.93	0.93	75
Wt avg	0.95	0.93	0.93	75

Aby 30/5/24

ANN using BP

```
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.max(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))

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 = hinp + bh
hlayer_act = sigmoid(hinp)
outinp = np.dot(hlayer_act, wout)
outinp = outinp + bout
output = sigmoid(outinp)

hiddengrad = derivatives_sigmoid(hlayer_act)
d_hiddenlayer = Eh * hiddengrad

print("Input:\n" + str(x))
print("Actual output:\n" + str(y))
print("Predicted output\n", output).
```

Output

Input

```
[[0.6667   1    ]
 [0.3333   0.5555]
 [1        0.6667]]
```

Actual Output

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

Predicted Output

```
[[0.78418718]
 [0.7253993 ]
 [0.78892728]]
```

30/5/24

Date
Page 19

ADA Boost

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_split
from sklearn.metrics

iris = load_iris()
X = iris.data
y = iris.target

X_train, X_test, y_train, y_test = train_test_
    split (x, y, size = 0.4)

adaboost_clf = AdaBoostClassifier (n_estimators = 30,
learning_rate = 1.0, random_state = 42)
adaboost_clf = fit (X_train, y_train)

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

Output

Accuracy : 0.96666

Week-8

```
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,
            c = colormap[y.Targets], s=40)
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,
  c = colormap[model.labels_], s=40)
plt.title('K-means Clustering')
```

plt.xlabel('Petal length')
plt.ylabel('Petal width')

Output

5

Implement Dimensionality reduction using PCA method:

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

from sklearn.datasets import load_breast_cancer
cancer = load_breast_cancer()
cancer.keys()
print(
```

OP: dict_keys(['DESCR', 'data', 'feature_names', 'target_names', 'target'])

```
print(cancer['DESCR'])

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

df.head().
```

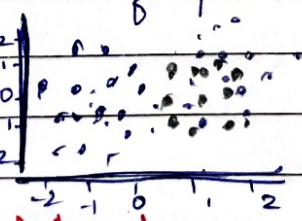
OP:

	mean radius	mean texture	mean perimetre	mean area	mean symmetry	... worst radius	worst ... area
0	17.99	10.38	122.80	1001.0	0.2419	25.38	17.33
1	20.57	17.77	132.90.	1326.0	0.1812	24.99	23.41
2	19.69.	21.25	130.00	1203.0	0.2069	23.57.	25.53
3	11.42	20.38	77.58	386.1	0.2877	14.91	26.50
4	20.29	14.34	135.10.	1297.0.	0.1809.	22.54	16.67

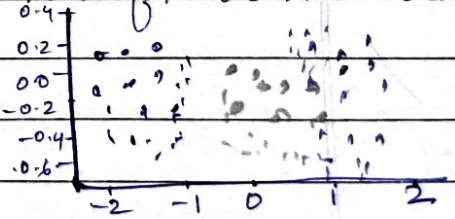
PCA Visualization

```
plt.figure(figsize=(8,6))
colors = ['r', 'g', 'b']
species_unique = df['species'].unique()
for target, color in zip(species_unique, colors):
    indices_to_keep = pca_df['species'] == target
    plt.scatter(pca_df.loc[indices_to_keep, 'Principal Comp i'],
                pca_df.loc[indices_to_keep, 'Principal component 2'],
                c=color,
                s=50)
plt.xlabel('Principal component 1')
plt.ylabel('Principal component 2')
plt.title(title)
plt.legend(species_unique)
plt.grid()
plt.show()

explained_variance = pca.explained_variance_ratio_
print(f"Explained variance for {title}: {
        explained_variance}")

perform_pca_and_plot(df, ['sepal_length',
'sepal_width'], 'PCA of Sepal Dimensions')

perform_pca_and_plot(df, ['petal_length',
'petal_width'], 'PCA of Petal Dimensions')
```

PCA of Sepal Dimensions



PCA of Petal Dimensions



30/5/24