21/5/24

Bafna Gold
Date: Page:

Week-1

Write a python program to import and export data using Pandas library function

(i) Reading data from CSV file

import pandas as pd

```
airbnb_data = pd.read_csv("/content/Airbnb_Open_data.csv")

airbnb_data.head()
```

- Output

	id	Name	neighbourhood	country	Service fee
0	1001254	Skylit Midtown Castle	Manhattan	United States	$ 98

(ii) #Reading data from URL

```
import pandas as pd

url = "https://archive.ics.uci.edu/ml/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

	sepal_length_in cm	sepal width in cm	petal length_in cm	petal width_in_cm	class
0	5.1	3.5	1.4	0.2	iris-setosa
1	4.9	3	1.4	0.2	iris-setosa
2	4.7	3.2	1.3	0.2	iris-setosa
3	4.6	3.1	1.5	0.2	iris-setosa
4	5	3.6	1.4	0.2	iris-setosa

Exporting dataframe to a CSV file

iris_data_to_CSV("cleaned iris data.csv")

Qua
2/1/21

28/3/24

Bafna Gold
Date: Page:

Week-2

1. ~~Dowload the data~~ look at the bigger picture
   - The california ~~de~~ census data is being used here.
   ~~import os~~
   - The important feature of the data are population, median income, Median housing price for each block group in california
   - The block group has a population between 600 to 3000

Selected Performance measure
→ Root mean Squared error
Check the assumption
→ An example for assumption can be the months worked on the algorithm be 6 month, etc

2. Get the data

# We are importing the python ~~libraries~~ like OS modules tarfile archive files and urllib (URL handling module)
import os
import tarfile
import urllib

We are forming the download root for the data set Path & a URL is created

```
Download_root = "https://raw.githubusercontent.com/ageron/
                handson-ml2/master/"
Housing_path = os.path.join("datasets", "01")
Housing_URL = Download_root + "datasets/housing/
              housing.tgz"

# We now define a function to fetch the housing data
  by giving it Housing_URL & Housing_Path as parameters

def fetch_housing_data(Housing_URL, housing_path=Hou
                                    sing_path):

    os.makedirs(name=housing_path, exist_ok=True)
    tgz_path = os.path.join(housing_path, "housing.tgz
    urllib.request.urlretrieve(housing_url,
                               filename=tgz_path
    housing_tgz = tarfile.open(name=tgz_path)
    housing_tgz.extractall(path=housing_path)

    housing_tgz.close()

# Call the fetch_housing_data() function

    fetch_housing_data()


# importing pandas library

    import pandas as pd
```

defining a function to load the housing data

```
def load_housing_data(housing_path=HOUSING_PATH):
    data_path = os.path.join(housing_path, "housing.csv")
    return pd.read_csv(data_path)
```

Loading the data into housing variable & displaying first 5 data rows

```
housing = load_housing_data()
housing.head()
```

Visualizing the data using histogram. We use matplotlib library.

```
import matplotlib.pyplot as plt

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

Creating a train & test set using numpy library

```
import numpy as np
def split_train_test(data, test_ratio=0.2):
    shuffled_indices = np.random.permutation(len(data))
    test_set_size = int(len(data)* test_ratio)
    test_indices = shuffled_indices[:test_set_size]
    train_indices = shuffled_indices[test_set_size:]
    train_indices = shuffled_indices[test_set_size:]
    return data.iloc[train_indices], data.iloc[
                                    test_indices]
train_set, test_set = split_train_test(data =
```

3. Discover & visualize, the data to gain insights
strat_train_set.shape, strat_test_set.shape

Visualizing
housing.plot(kind='scatter', x='longitude',
y='latitude'), alpha=0.1)

plt.show()

Correlation
corr_matrix = housing.corr()
corr_matrix['median_house_value']
sort_values(ascending=False)
from pandas.plotting import scatter_ma
attributes = ['median_house_value',
'median_income']
scatter_matrix(frame=housing[attribute
figsize=(12,8))

plt.show()
housing['rooms_per_household'] =
housing['total_rooms']/housing['
household']

4. Prepare the data for machine learning algorithms
from sklearn.impute import SimpleImpu
imputer = SimpleImputer(strategy=
'median')
housing_num = housing.drop("ocean_
proximity", imputer.fit(housing_num)

imputer.statistics
housing_num.median().values

- from sklearn.preprocessing import ordinal Encoder ordinal_encoder = OrdinalEncoder()
  ordinal_encoder.categories_
- attr_adder = CombinedAttributesAdder(add_bedroom, per_room=False)
  housing_extra_attribs = attr_adder.transform(housing.values)
- from sklearn.compose import column transform num_attribs = housing_num.columns.tolist()

5. Select and train a model:

- from sklearn.linear_model import LinearRegression
  lin_reg = LinearRegression()
  lin_reg.fit(X = housing_prepared, y = housing_labels)
- housing_prediction = 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 and Maintain your system

4/4/24

Week-4

Simple linear regression

Code:

```
def estimate_coef(x, y):
    n = np.size(x)
    m_x = np.mean(x)
    m_y = np.mean(y)

    SS_xy = np.sum(y*x) - n*m_y*m_x
    SS_xx = np.sum(x*x) - n*m_x*m_x

    b_1 = SS_xy / SS_xx
    b_0 = m_y - b_1*m_x

    return (b_0, b_1)

def plot_regression_line(x, y, b):
    plt.scatter(x, y, color = "m", marker = "o", s = 30)

    y_pred = b[0] + b[1]*x

    plt.plot(x, y_pred, color = "g")

    plt.xlabel('x')
    plt.ylabel('y')

def main():
    x = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
y = np.array([1, 3, 2, 5, 7, 8, 8, 9, 10, 12])

b = estimate_coef(x, y)
print("Estimated coefficients:\nb_0 = {}\
\nb_1 = {}".format(b)
```

Steps

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from pandas.core.common import
from sklearn.linear_model import LinearRegression

df_sal = pd.read_csv('/content/Salary_Data.csv')
df_sal.head()

df_sal.describe()

plt.title('Salary distribution Plot')
sns.distplot(df_sal['Salary'])
plt.show()

plt.scatter(df_sal['Years Experience'],
            df_sal['Salary'], color='light coral')
plt.title('Salary Vs experience')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.box(False)
plt.show()
```

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

x_train, x_test, y_train, y_test =
train_test_split(x, y, test_size = 0.2, random
_state = 0)

regressor = regressor.predict(x_test)
regressor.fit(X_train, y_train)

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

plt.scatter(X_train, y_train, color='light
coral')
plt.plot(X_train, y_pred_train, color='firebrick')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.legend(['X_train / Pred(y_test)', 'X_train
/y_train'], title = 'Sal/Exp',
loc='best', facecolor='White')
plt.box(False)
plt.show()

plt.scatter(X_test, y_test, color='light_coral')
plt.plot(X_train, y_pred_train, color=
'firebrick')

plt.show()
print(f'Coefficient : {regressor.coef_}')
print(f'Intercept : {regressor.intercept_}')
```

Multiple - linear Regression

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import linear regressor

df_start = pd.read_csv('/content/50_Startups (2).c
df_start.head()

df_start.describe()

plt.title('Profit distribution plot')
sns.xlabel('R&D spend')
plt.ylabel('profit')
plt.box(False)
plt.show()

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

ColumnTransformer(transformers=[('encoder',
 OneHotEncoder(),[3])], remainder='pass
 through')

X_train, X_test, y_train, y_test = train_
    test_split(X, y, test_size = 0.2,
    random_state)
```

```
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
    (y_test, reshape(len(y_test), 1)
```

~~result~~

v'))

Week-4

Use an appropriate data set for building the decision and apply this knowledge to classify a new sample

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
import seaborn as sns

iris_data = load_iris()

print(iris_data.DESCR)
X = iris_data.data
Y = iris_data.target

print("Shape of X:", X.shape)
print("Shape of Y:", Y.shape)

from sklearn.model_selection import
train_test_split
X_train, X_test, y_train, y_test = train
_test_split(X, y, test_size = 0.33, random

from sklearn.tree import DecisionTreeClassifi
tree_model = DecisionTreeClassifier()

treemodel.fit(X_train, y_train)
DecisionTreeClassifier()
```

```
from sklearn.tree import DecisionTree
classifier, plot_tree
import matplotlib
pyplot as plt.

df = DecisionTreeClassifier(
df.fit(x, y)

plt.figure(fig_size = (12,8))
plot_tree(df, filled = True, feature_name=
  feature_name, class_names = iris.data.target
plt.show()
```

Output :

pet. width (cm) $\alpha = 0.8$
gini = 0.667
samples = 150
values = [50, 50, 50]
class = setosa

gini = 0.0
samples = 50
values [50, 0, 0]
class = setosa

petal width (cm) $\alpha = 1.75$
gini = 0.5
samples = 100
values = [0, 50, 50]
class = versicolor

petal length (cm) $\alpha = 4.95$
gini = 0.168
samples = 54
value = [0, 49, 5]
class = versicolor

petal length (cm) $\alpha = 4.85$
gini = 0.043
Samples = 46
value = [0, 1, 45]
class = virginica

```
from sklearn.model_selection import
cross_val_score

scores = cross_val_score(clf, X, y, cv=5)
accuracy = scores.mean()
print("Mean Accuracy:", accuracy)
```

Output

Mean Accuracy : 0.9666

25-4-2024

Bafna Gold
Date: Page:

25/4/24

Week 5

5. Build logistic Regression model for a given dataset

```
import pandas as pd
from matplotlib import pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

df = pd.read_csv("insurance_data.csv")
print(df.head())

X_train, X_test, y_train, y_test = Train_test_
    split(df[['age']]&
        df.bought_insurance, test_size=0.3
print("X_test:")
print(X_test)

model = LogisticRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print(y_pred)
print(model.predict_proba(X_test))
print(model.score(X_test, y_test))
```

Output:

Accuracy = 0.5

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

```
def predi(age):
    z = 0.042 * age - 1.53
    y = sigmoid(z)
    return y

print(predi(35))
print(predi(43))
```

Output:

Prediction : array([1, 0, 1, 0, 0, 0, 0, 1, 0])

score : 0.8888

linear Reg score : 0.5841371

Predictions : 0.485

0.5685

9/5/24 Week 6

Q Build KNN classification model for a dataset

Code:

```
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
import matplotlib.pyplot as plt
import seaborn as sns

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

iris_df = pd.DataFrame(X, columns=iris.feature_names)
iris_df['target'] = y

sns.pairplot(iris_df, hue='target')
plt.show()

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

k = 3
knn_classifier = KNeighborsClassifier(n_neighbors=k)

knn_classifier.fit(X_train, y_train)
```

KNeighborsClassifier
KNeighborsClassifier(n_neighbors=3)

y_pred = knn_classifier.predict(X_test)

accuracy = accuracy_score
print("Accuracy", accuracy)

Accuracy: 0.9833

Q. Build support vector machine model for a given dataset

Code:
~~import pandas as pd~~
~~from sklearn.~~

from sklearn.svm import SVC
model = SVC(kernel='linear', random_state=0,C=1
model.fit(x_train, y_train)
y_2 = model.predict(X_train)

from sklearn.metrics import accuracy_score
score = accuracy_score(y_pred, y_test
print(f"Testing accuracy: {score}")
score2 = accuracy_score(y_2, y_train)
print(f"Training accuracy: {score2}")
Output:

Training Accuracy = 1.0
Training Accuracy = 1.0

23/5/24 Week-7

Q Build artificial neural network model with back propagation on a given dataset

```
import numpy as np
x = np.array([[2,9],[1,5],[3,6]], dtype=float)
y = np.array([[92],[86],[89]], dtype=float)
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))
wout = np.random.uniform(size=(hiddenlayer_neurons, outputlayer_neurons))
bout = bh = np.random.uniform(size=(1, outputlayer_neurons))
def sigmoid(x):
    return 1/(1+np.exp(-x))

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

for i in range(epoch):
    hinpl = np.dot(x, wh)
    hinp = hinpl + bh
    hlayer_act = sigmoid(hinp)
```

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_act)
d_hiddenlayer = EH * hiddengrad
wout += hlayer_act.T.dot(d_hiddenlayer) *
print("Input: \n" + str(x))
print("Actual output: \n" + str(y))
print("Predicted output: \n", output)

Output:

[[0.66666667 1. 0.3333333 0.66666667]]

[[0.92]
[0.86]
[0.89]]

Predicted output:

[[0.8913061]
[0.88551981]
[0.88962179]]

Ewa
30/5/24

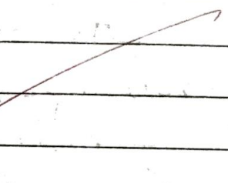
lab-7

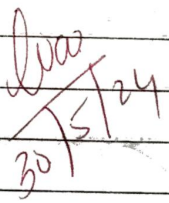
Artificial Neural Network:

```
import numpy as np
X = np.array([[2,9],[1,5],[3,6]], type=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 = 3
outputneurons = 1
Wh = np.random.uniform(size=(inputlayer_
neurons, hiddenlayer_neurons))
bh = np.random.uniform(size=(1, hiddenlayer))
wout = np.random.uniform(size=(hidden
                               layer_neuron, output_
                               neurons))
bout = np.random.uniform(size=(1, output
                               _neurons))

# Sigmoid function
def sigmoid(x)
    return 1/(1+np.exp(-x))
Eb = y-output
outgrad = derivation_sigmoid(output)
d_output = Eo * output
```

```
EH = d_output.dot(wout.T)

hiddenpred = derivatives_sigmoid(hlayer)
d_hiddenlayer = EH * hiddenpred
wout += hlayer.act.T.dot(d_output)
        * lr
print("Input=" + str(x))
print("actual output" + str(y))
print("predicted output", output)
```

actual output = [0.92] [0.86] [0.89]
predicted output [0.84 0.85] [0.87]





Bafna Gold
Date: Page:

Lab - 8
Program

9a. Random Forest:

```
import numpy as np
import random as rd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report

data = pd.read_csv('/content/spam data.csv')
X = data['Body']
Y = data['Label']

vectorizer = CountVectorizer()
X = vectorizer.fit_transform(X)

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

print(classification_report(y_test, y_pred))
accuracy : 0.9731
```

lab-9

9b Adaboost

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

df = Pd.read_csv("/content/iris.csv");
df.head()

y = df['species'], axis=1)

from sklearn.model_selection import
train_test_split
X_train, X_test, y_train, y_test = train_
    X, y, test_size = 0.3, test_split=0)

from sklearn.ensemble import
AdaBoostClassification()
dt_pred = adb.predict(X_test)
y2 = adb.predict(X_train)

from sklearn.metrics import
accuracy
acc_score = accuracy.score(
    y_pred, ytest)
print(accuracy_score)
```

=0.977

3-2-22

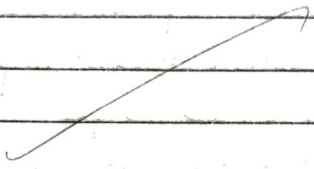
30/9/24 lab-10

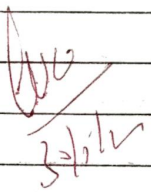
K-Means algorithms

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)

plt.figure(figsize=(14,14))
colormap = np.array(['red', 'lime', 'black'])

plt.subplot(2,2,1)
plt.scatter(X.Petal_length, X.Pental_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')

plt.show()
```





30/5/24 Lab-10

Principle component analysis

```
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()

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

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
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')