INDEX

Name ROSHINI. R Std Sec

Roll No. Subject School/College

School/College Tel. No. Parents Tel. No.

21/3/24

Date __/__/__

Page _____

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

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

Output:

id	name	host_id	host_name	neighbourhood_group	latitude	
2265	Zen-east	2466	Paddy	null	78702	97.812
5245	Austin	2466	Paddy	null	78729	-98.98

Date
Page

Reading data from URL

```
url = "http://archive.ics.uci.edu/ml/machine-learning-databases/
      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_len_in_cm	sepal_width	petal_length	petal_wid
5.1	6.0	1.4	0.1
5.9	7.0	1.3	0.2
2.8	4.8	1.4	0.1

Exporting dataframe to a csv file

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

Output:

	sepal_length	sepal_width	petal_length	petal_widt
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.3	0.2
2	4.7	3.2	1.3	0.2

28/3/24

Date __/__/__
Page _____

Get the data

```
import os
import tarfile
import urllib

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

housing.head()
```

Discover and visualize the data

```
strat_train_set.shape, strat_test_set.shape
strat_test_set.reset_index().to_feather
                (fname:"data/01/strat_test_set.f")
housing = strat_train_set.copy(); housing.
                                    shape
housing.plot(kind='scatter', x='longitude',
             y='latitude')
plt.show()
```

Date / /
Page

Prepare the data for machine learning algorithm

```
housing = strat_train_set.drop("median_house_value", axis=1)
housing_labels = strat_train_set["median_house_value"].copy()
housing.shape, housing_labels.shape
imputer = SimpleImputer(strategy="median")
imputer.fit(housing_num)
imputer.statistics_
```

Select and train a model

```
lin_reg = LinearRegression()
lin_reg.fit(x=housing_prepared, y=housing_labels)
some_data = housing.iloc[:5]
housing_predictions = lin_reg.predict(housing_prepared)
lin_rmse = np.sqrt(lin_mse)
lin_rmse
```

Fine-Tune Your model

```
from sklearn.model_selection import GridSearchCV
param_grid = {'n_estimators': [3, 10, 30],
              'max_features': [2, 4, 6, 8]}
forest_reg = RandomForestRegressor()
grid_search.fit(x=housing_prepared,
                y=housing_labels)
cvres = grid_search.cv_results_
```

Date / /
Page

```
x_test = strat_test_set.drop("median_house_value")
y_test = strat_test_set["median_house_value"].copy()
```

O/p:

Output:
The model is trained using the dataset

Simple Linear Regression

Import libraries
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection
```

Import Data
```
df_sal = pd.read_csv('/content/Salary_dat
                      csv/')
```

Analyze data

Describe
```
df_sal.describe()
```

Distribution
```
sns.distplot(df_sal['salary'])
```

Relationship between salary and experience
```
plt.scatter(df_sal['yearofexperience'],
            df_sal['salary'])
plt.title('salary vs experience')
plt.show()
```

Split data
split into independent / dependent variables
```
x = df_sal.iloc[:, :1]
y = df_sal.iloc[:, 1:]
```

Date __/__/__
Page _____

Visualize predictions

Prediction on training sets
```
plt.scatter (x_train, y_train, color, 'lightcoral')
plt.plot ('x_train, y_pred_train')
plt.xlabel ('years of experience')
plt.ylabel ('salary')
```

Prediction on test set
```
plt.scatter(x_test, y_test)
```

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

O/P:

Output:
coefficient of regression: 8

4/4/2

Date / /
Page

Multiple linear Regression

```
Import libraries
import pandas as pd
import numpy as np
import seaborn as sns

Import data
df_start = pd.read_csv('/content/50_startup.csv')

Analyze data

Describe
df_data.describe()

Distribution
plt.title('Profit Distribution plot')
plt.show()

Split into independent/dependent
X = df_start.iloc[:, :-1]
y = df_start.iloc[:, 1]

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)
```

18/4/24

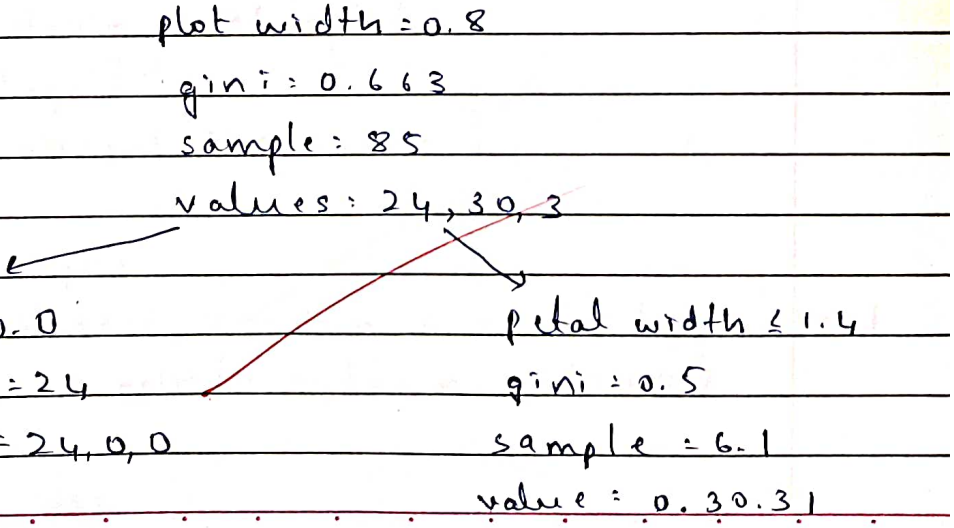
Date __/__/__
Page _____

Week-4

Use an appropriate data set for building the decision tree and apply this knowledge to classify

```
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.tree import decisiontreeclassification
from sklearn import tree
iris = load_iris()
y = iris.data
y = iris.length
from sklearn.model_selection import train
y_train = y_test = train_test_split(y, y,
                                    test_size=0.43)
tree_model = decisiontreeclassifier
tree_model = tree_model.fit(x_train, y_train)
plt.figure(figsize=(15,20))
tree.plot_tree(tree_model, filled=True,
               name=iris.feature_name);
```

Output:



```
y_pred = true_model.predict(x_test)
from sklearn metrices import accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy", accuracy)
```

Output:

Accuracy = 0.96923

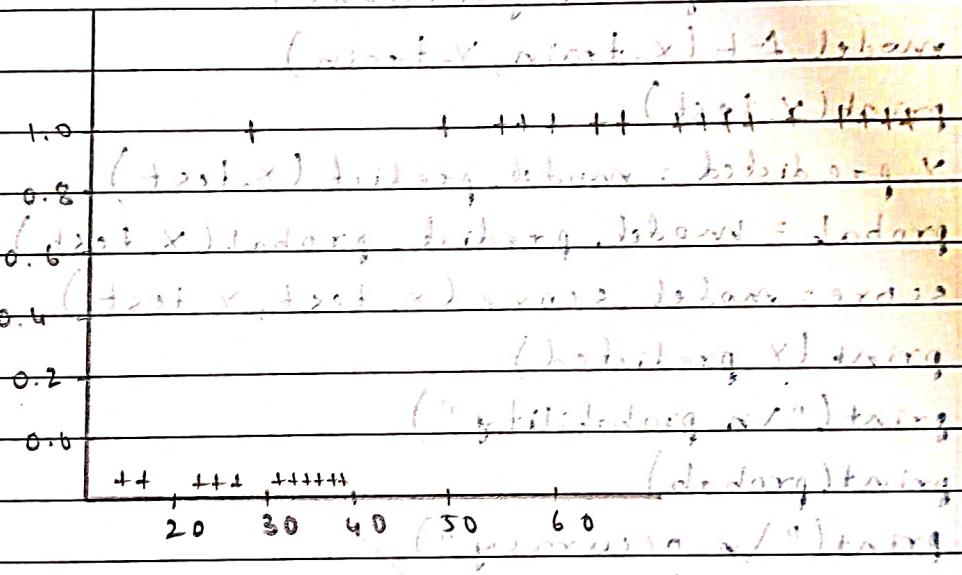
25/4/24

Date / /

Page

Logistic Regression

```
import pandas as pd
from matplotlib import pyplot as plt
%matplotlib in ln
df = pd.read_csv("insurance_data.csv")
df.head()
plt.scatter(df.age, df.bought_insurance,
            marker='+', colour='red')
from sklearn.model_selection import
                    train_test_split
```




```
from sklearn.model_selection import
            train_test_split
X = df[['age']]
Y = df['bought_insurance']

Perform split
x_train, x_test, y_train, y_test = train_test_
            split(x, y, train_test=0.8)

Display x_test
```

```
print(x_test)
```

	age
2	47
4	61
5	56
21	26
22	40
17	58

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(x_train, y_train)
print(x_test)
y_predicted = model.predict(x_test)
probab = model.predict_probab(x_test)
score = model.score(x_test, y_test)
print(y_predicted)
print("\n probability")
print(probab)
print("\n accuracy")
print(score)
```

x_test:

	age
9	61
23	45
4	46
24	50
15	55
1	25

predicted values : [1 1 1 1 0 1 1 0]

probabilities : [[0.09781523 0.90218477]
[0.40313939 0.59686061]
[0.37545927 0.62454073]
[0.27608642 0.72391358]
[0.86924175 0.13075825]]

accuracy 1.0

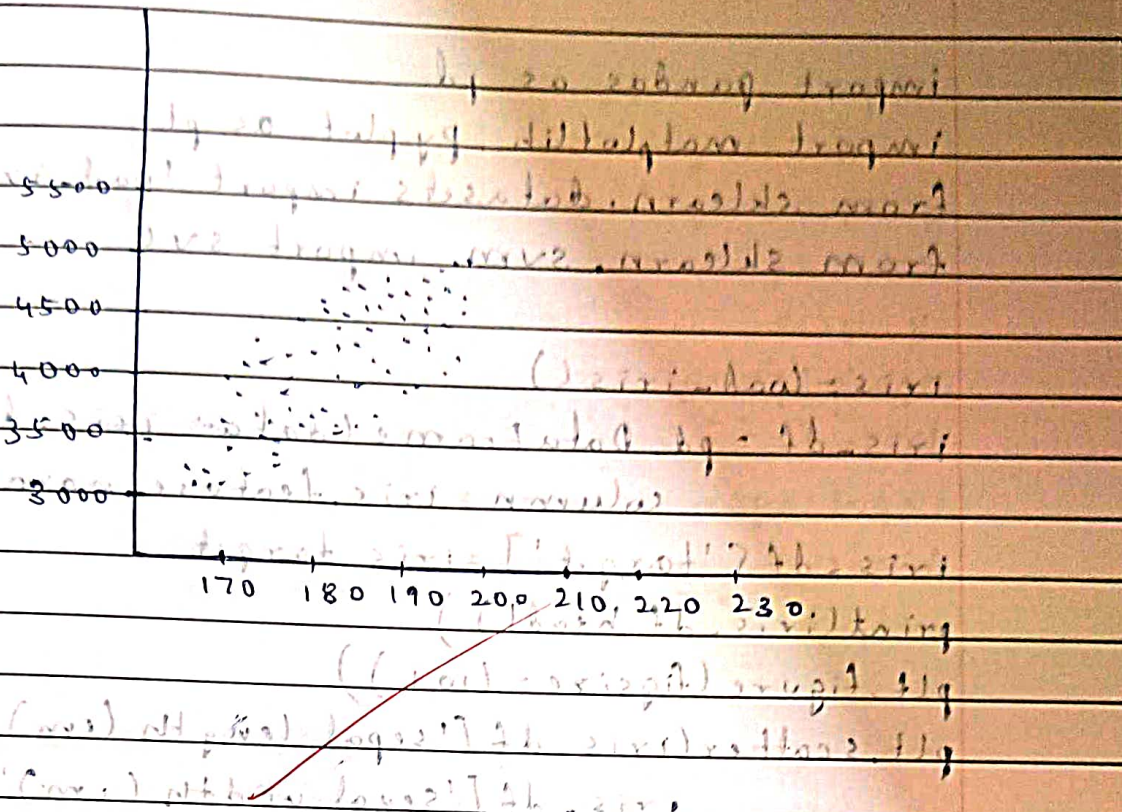
Date / /
Page / /

KNN

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

df=pd.read_csv("https://raw.githubuser.com
            /allisonhorst/palmerpenguins/master/penguins

print(df.head())

sns.scatterplot(x="flipper_length_mm",
    y="body_mass_g", hue="species", data=df)
plt.title("Penguin species Classification")
plt.xlabel("Flipper length(mm)")
plt.ylabel("Body mass(g)")
plt.show()

df.dropna(inplace=True)

X=df[['flipper_length_mm','body_mass_g']]
y=df['species']
X_train, X_test, y_train= test_size=0.2

KNN=KNeighborClassifier(n_neighbor=3)
KNN.fit(x_train, y_train)
y_pred=KNN.predict(x_test)
print("Accuracy", accuracy)
```

Date __/__/__
Page _____

Output:

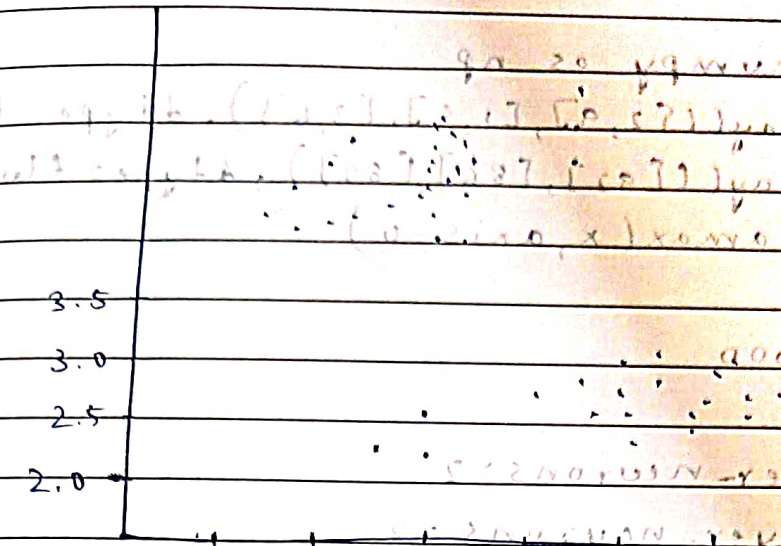


SVM

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.svm import SVC

iris = load_iris()
iris_df = pd.DataFrame(data = iris.data,
                       column = iris.feature_names)
iris_df['target'] = iris.target
print(iris_df.head())
plt.figure(figsize = (10,6))
plt.scatter(iris_df['sepal length (cm)'],
            iris_df['sepal width (cm)'],
            c = iris_df['target'),
            cmap = 'viridis')

plt.xlabel('sepal length (cm)')
plt.ylabel('sepal width (cm)')
plt.title('Iris dataset')

x_train, x_test, y_train, y_test = train_test
                                   split(iris.data, iris.tar
                                   test_size = 0.3,
                                   random_state = 42)

svm_classifier = SVC(kernel = 'linear')
svm_classifier.fit(x_train, y_train)
y_pred = svm_classifier.predict(x_test)

print("Accuracy of SVM classifier",
                                   accuracy)
```

Date __/__/__
Page _____

Output:



23/5/24

Date / /
Page / /

Implementation of 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.amax(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, output_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 = hinp1+bh
hlayer_act = sigmoid(hinp)
outinp1 = np.dot(hlayer_act, wout)
```

```
outinp = outinp1 + bout1
output = sigmoid(outinp)

EO = y - output
outgrad = derivatives_sigmoid(output)
d_output = EO * outgrad
EH = d_output.dot(wout1.T)

hiddengrad = derivatives_sigmoid(hlayer_act)
d_hiddenlayer = EH * hiddengrad

wout1 = hlayer_act.T.dot(d_output) * lr
wh1 = x.T.dot(d_hiddenlayer) * lr
print("Input: \n" + str(x))
print("Actual Output: \n" + str(y))
print("Predicted output: \n", output)
```

Output:

Input:

```
[[0.66667 1.     ]
 [0.33333 0.55556]
 [1.      0.66667]]
```

Actual output:

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

Predicted output:

```
[[0.78415718]
 [0.76533937]
 [0.78892728]]
```

Date
Page

Random Forest Algorithm

```
from sklearn.datasets import load_iris
from sklearn.model_selection import
        train_test_split
from sklearn.ensemble import
        RandomForestClassifier
from sklearn.metrics import accuracy_
        score, classification_repor

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

x_train, x_test, y_train, y_test = train_te
    split(x, y, test_size = 0.5, random_stat

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

y_pred = rf_classifier.predict(x_test)

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

Accuracy: 0.933333

```
classification_rep = classification_report
        (y_test, y_pred)
print("\n Classification Report:\n",
        classification_rep)
```

Date __/__/__
Page _____

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
weighted avg	0.95	0.93	0.93	75

Date
Page

Adaboost

```
from sklearn.datasets import load_iris
from sklearn.model_selection import
                    train_test_split
from sklearn.ensemble import
                    AdaboostClassifier
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 = 0.4, random_state

adaboost_clf = AdaBoostClassifier(
            n_estimators = 30, learning_rate = 1.
            random_state = 42)
adaboost_clf.fit(x_train, y_train)

y_pred = adaboost_clf.predict(x_test)

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

Output:

Accuracy: 0.96666667

23/5/24