Name Prajwal Bhagat Sub.

Std.: Div. Roll No.

Telephone No. E-mail ID.

Blood Group. Birth Day.

Sr.No.	Title	Page No.	Sign./Remarks
1.	Import & exporting data		
2.	End to End ml project		
3.	Linear Regression multiple Linear Regression		
4.	Decision tree		
5.	Logistic Regression. KNN		
6.	K means SVM PCA		
7	Ensemble learning Random forest Boosting.		

NL 30/5/24 Incomplete knowledge in concepts.

PAGE NO :
DATE :

21/3/24

Import and Export Pandas library further

```
import pandas as pd

col_names = ["sepal_length_in_cm", "sepal_width_in_cm", "petal_length_in_cm",
             "petal_width_in_cm", "class"]

data = pd.read_csv("C:\Users\Admin\Desktop\iris\iris.data");

data.columns = col_names.
```

Output:

sepal length in cms	sepal width	petal length	petal width
4.9	3.0	1.4	0.2
4.7	3.2	1.3	0.2

Class
Iris-setosa
Iris-setosa

27/3/24

End to End ML Project.

1. Problem formulation
   formulate the problem by considering what input has to be given what is expected output and features available.

2. get the data
   * import os
     - provide functions for interacting with os
   * import URL lib
     provides functions for fetching URL's

   import pandas as pd
   provide functions to work and manipulate data sets.

   housing.head() => returns top 5 rows of file

   housing.info() => gives the count of non null values in each attribute.

   * housing.describe() gives the statistical measures such as count mean, std, min, max and other attribute features.

   split_train_test()
   it divides the dataset into train set and test set the ratio can be specified.

train_set.shape
gives the shape of train set

housing['id'].value_counts()
gives the count of values.

3. Discover and visualize the data to gain insights
* train_test_set.shape()
* we plot scatterplot by using
housing.plot(kind='scatter', x-lon, y-lat)
plt.show()

* using scattermatrix plot correlation.
scatter_matrix(frame = housing[attribute], fig_size=(12,7))

4. prepare data for ML algorithms.

→ Data cleaning
* get rid of whole attributes
housing.drop('total_bedrooms', axis=1)
* set missing values to some value
zero, mean, median.

→ Handling text and categorial attributes

→ Custom Transformers
→ we use fit, transform

→ feature Scaling
min-max scaler & standardization.

PAGE NO :
DATE :

5. Select a train model.
we use linear regression model
Decision tree regression - it is a machine learning algo used for regression tasks where the goal is to predict a continuous target variable

6. Tune your model
→ Grid search CV is a model provided by Scikit learn library in python for hyper parameter tuning of machine learning models.

7. launch, monitor & maintain your system
we can automate the process by
→ collecting fresh data regularly & labelling it
→ writing script to train models & fine tune the hyper parameters
→ writing script to evaluate model.

PAGE NO:
DATE:

Implementation of Linear Regression

```
import numpy as np
import matplotlib as plt

def estimate_coef(x,y)
    n = np.size(x)
    mx = np.mean(x)
    my = np.mean(y)
SS_xy = np.sum(y*x) - n*my*mx
    b_1 = SS_xy/SS_xx
    b_0 = my - b_1*mx
    return (b_0,b_1)

def plot_regression_line(x,y,b)
    plt.scatter(x,y, color = "m", mark = 0.5-10)
    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,...9])
    y = np.array([1,3,4,...12])
    b = estimate_coef(x,y)
    print(b)
    plot_regression_line(x,y,b)
```

Output:

(b_0, b_1) : (1.2363, ... 1.169691...)

PAGE NO:
DATE:

~~matlo~~ multiple linear Regression

```
from sklearn model_selection import train_test_split
import matplotlib as plt
import numpy as np.
from sklearn import datasets linear_model, metrics

data_url = "http://lib.stat.cmu.edu/datasets/boston"
raw_df = pd.read_csv(data_url, skiprows=22,
                     header=None)

x = np.stack([raw_df.values[::,2,1], raw_df.value[::,
y = raw_df.values[1::2,2].

reg = linear_model.LinearRegression()
reg.fit(x_train, y_train)

print("Coefficients:", reg.self)
print("variance score = {}".format(reg.score(x_test,
                                    y_test))

plt.style.use("five thirty eight")

~~print("co~~
plt.scatter(reg.predict(x_train), reg.predict(x_train)
            y_train, color = "green", s=10,
            label = "Train-data")

plt.scatter(reg.predict(x_test), reg.predict(x_test)
            y_test, color
```

PAGE NO :
DATE :

plt.hlines(y=0, xmin=0, xmax=50, linewidth=2)

plt.legend(loc="upper right")
plt.title("Residual error")
plt.show().

PAGE NO :
DATE :

Implementation of ID3.

```
import numpy as np
import pandas as pd
eps = np.info(float).eps
from numpy import log2 as log.

from google.colab import drive
drive.mount('/content/drive')
path = '/drive/MyDrive/Colab Notebooks/playtennis.csv'
df = pd.read_csv(path)

def find_entropy(df):
    target = df.keys()[-1]
    entropy = 0
    values = df[target].unique()

    for value in values:
        fraction = df[target].value_counts()[value]/len(df[target])
        entropy += -fraction * np.log2(fraction)

    return entropy

def average_information(df, attribute):
    target = df.keys()[-1]
    target_variables = df[target].unique()
    variables = df[attribute].unique()
    entropy = 0
    for variable in variables:
        entropy = 0
        for target_variable in target_variables:
            num = len(df[attribute][df[attribute]==
```

variable] [df['target'] == target[variable])

den = len(df [attribute] [df[attribute] == variable])
fraction = num / (den + eps)
entropy += - fraction * log(fraction + eps)
fraction 2 = den / len(df)
entropy 2 += fraction 2 * entropy

return abs(entropy)

tree = build Tree (df)

import pprint
pp print = pprint.pprint (tree)

output = { 'outlook' : { 'overcast' = 'yes',
'rainy' : { 'windy' : { False: 'yes';
True : 'no' } },
'sunny': { 'humidity' : { 'high' : 'no',
'normal' : 'yes' } } } }

9/8/24

PAGE NO:
DATE:

Implementation of Logistic Regression

```
import pandas as pd
import matplotlib as plt
% matplotlib inline

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

from sklearn.model_selection import train_test_split

plt.scatter(df.age, df.bought_insurance, marker='+',
            color='red')

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

print(X_test)

y_predicted = model.predict(X_test)
model.predict_proba(X_test)
model.score(X_test, y_test)
print(y_predicted)
print(X_test)
```

PAGE NO:
DATE:

```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_train, y_train)
print("Coefficient (m):", model.coef_)
print("Intercept (b):", model.intercept_)

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

def prediction_fun(age):
    m = 0.042
    b = -1.53

    Z = m*age + b
    y = sigmoid(b)

    return y
```

N.P
9/5/24

PAGE NO :
DATE :

Implementation of KNN

```
import numpy as np
import pandas as pd.

dataset = pd.read_csv('/content/drive/mydrive/
                       ML-LAB/iris.csv')

dataset.shape

dataset.head()

dataset.describe()

dataset.groupby('species').size()

feature_columns = ['sepal length cm', 'sepal width cm',
    'petal length cm', 'petal width cm'].

X = dataset[feature_columns].value
y = dataset['species'].values.

from sklearn.preprocessing import Label encoder
Le = Label encoder()
y = Le.fit_transform(y)

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test
    split(X, y, test_size = 0.2, random_state = 0)
```

PAGE NO :
DATE :

import matplotlib.pyplot
import seaborn as sns

from sklearn.neighbours import KNeighboursClassifier

from sklearn.metrics import confusion_matrix, accuracy_score

from sklearn.model_selection import cross_val_score

classifier = KNeighboursClassifier(n_neighbours=3)

classifier.fit(x_train, y_train)

y_pred = classifier.predict(x_test)

accuracy = accuracy_score(y_test, y_pred)*100

print(accuracy)

9/5/24

PAGE NO :
DATE :

SVM

```
from sklearn.datasets import load_breast_cancer
import matplotlib.pyplot as plt
from sklearn.inspection import DecisionBoundaryDisplay
from sklearn.svm import SVC

cancer = load_breast_cancer()
X = cancer.data[:, :2]
y = cancer.target

svm = SVC(kernel="rbf", gamma=0.5, C=1.0)
svm.fit(X, y)

DecisionBoundaryDisplay.from_estimator(
    svm,
    X,
    response_method="predict",
    cmap=plt.cm.Spectral,
    alpha=0.8,
    xlabel=cancer.feature_names[0],
    ylabel=cancer.feature_names[1],
)
plt.scatter(X[:, 0], X[:, 1], c=y, s=20, edgecolors="k")
plt.show()
```

PAGE NO :
DATE :

PCA

```
import pandas as pd
import numpy as np
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

data = pd.read_csv("____")
Scaling = StandardScaler()
Scaling.fit(data)

Scaled_data = Scaling.transform(data)
principal = PCA(n_components = 3)
principal.fit(Scaled_data)
x = principal.transform(Scaled_data)

plt.figure(figsize=(10,10))
plt.scatter(x[:,0], x[:,1], c=data['target'],
plt.xlabel('PCA')
plt.ylabel('PC1')
```

Kay means

```
import pandas as pd
data = pd.read_csv("____.")
data.head()
import numpy as np
from sklearn.datasets import load_iris
from sklearn.cluster import Kmeans

Y, j = load_iris(return_x_y = True)
k means = Kmeans(n_Cluster = 3, random_state = 2)
Kmean.fit(x)
pred = Kmean.fit_predict(x)
pred
```

PAGE NO :
DATE :

Ensemble Boosting

```
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=42)

adaboost_clf = AdaBoostClassifier(n_estimators=50, learning_rate=1.0, 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)
```

PAGE NO :
DATE :

Ensemble Random Forest

import numpy as np
import pada as pd

import matplotlib.pyplot as plt
import Seaborn as sns

import SKlearn
from SKlearn import datasets
from SKlearn.model_selection import train_test_split

from SKlearn.ensemble import RandomForestClassifier.

from SKlearn.metrics import accuracy_score, confusion_matrix, classification_report

iris = datasets.load_iris()
type(iris)

iris.target_names.
iris.feature_names.
iris.target.

iris_data = pd.DataFrame({
'sepal length': iris.data[:,0],
'sepal width': iris.data[:,1],
'petal length': iris.data[:,2],
'petal width': iris.data[:,3],
'species': iris.target
})

iris_data.head()

x = iris_data.iloc[:, :-1].values
y = iris_data.iloc[:, -1].values
x_train, x_test, y_train, y_test =
train_test_split(x, y, test_size=0.3?, random_state
model = RandomClassifier()

model.fit(x_train, y_train)

y_pred = model.predict(x_test)
accuracy_score(y_test, y_pred)

model11 = RandomForestClassifier(n_estimators=100)
model11.fit(x_train, y_train)

feature_imp = pd.Series(data=model11.feature_importances_,
index=iris.feature_names)

feature_imp = feature_imp.sort_values(ascending=False)
feature_imp

sns.barplot(x=feature_imp, y=feature_imp.index)
plt.xlabel('Feature Importance Score')
plt.ylabel('Features')
plt.title("Visualizing Important Features")
plt.show()

20/1/21