[illegible]

## End to End ML project

1. Get data :-

os. path . join → specifies the directory where the extracted data shd be downloaded and stored.

os. makedirs → makes directory.

* describe() and info() function to analyse data

* Median-house-value is capped.

* housing . hist ( bins = 50 , figsize (20, 15) )
- histogram is ploted which is used for univariate analysis & bins are used to specify the width of each group

* split - train - test ( )
It divides the dataset into train set & test set , test ratio can also be specified to divide the dataset

* train - set . shape
gives shape of train set (columns & rows)

* housing [id] . value counts ( )
→ frequency of id columns

2. Discove
* strat
* we

3. prepa
a) Data (

b) Hand
c) cust
d) featur

4) selec
se

5) Fine
Gr
sc
tun

6). Lau
* a
- colle
- co

- W

2. Discover & visualize the Data to gain insight
* strat train set. shape()
* We plot scatter plot by using
  housing.plot(kind='scatter', x='longitude', y='latitude'). plt.show()

3. prepare the data for Machine learning
a) Data Cleaning
  Get rid of whole attribute (feature)
  housing.drop('total-bedrooms', axis=1)
b) Handling Text & categorical Attributes
c) Custom Transformers
d) Feature scaling

4) Select & train a Model
  Linear Regression, Decision Tree Regussor.

5) Fine tune your model
  Grid search CV is a method provided by scikit-learn library in python for hyperparameter tuning of ML models.

6). Launch, Monitor & Maintain your system.
* automate by:-
- collecting fresh data regularly & labelling it
- writing script to train modle & fine tune the hyperparameters
- writing script to evalute the model

13/11/24

## python Implementation of Linear Regression

```
import numpy as np
import matplotlib.pyplot as plt

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):
    plot . scatter (x, y, color = "m", marker = "o",
    y_pred = b[0] + b[1]*x    s = 30)
    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, 2, .. 12])

    b = estimate_coef (x, y)
    print (b)
```

plot_r

Output :-

(b_0, b

→ Multiple

from sklea
  split
import n
import n
from skle

data_wr

raw_dt

x = m

y = r

x_tra

reg =
  reg

prin

```
plot_regression_line (x, y, b)
```

Output :-

```
(b_0 , b_1) = (1.2363 ..., 1.16969 ...)
```

→ Multiple linear Regression

```
from sklearn.model selection import train_test
    split
import matplotlib.pyplot 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, sep = "\s+",
            skiprows = 22, header = None)

x = np.hstack (craw_df.values [::2, :],
            raw_df.values [1::2, :2]])
y = rawdf.values [1::2, 2]

x_train , x_test , y_train , y_test = train_test_
            split (x, y, test_size = 0.4,
                random_state = 1)

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

print (" Coefficient :", reg.coef_)
print ("variance score: {}". format
       (reg.score (x_test, y_test)))

plt. style. use (' five thirtyeight ')

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 = "blue" , s = 10 , label = "test data')

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

plt. legend (loc = " upper right ' )

plt. title (" Residual errors ")

plt. show ()
```

25/4/24

25/4/24

Deci

01) Readi

02) Im

def

Ave

Inf

25/4/24

Decision tree ID3 implementation using play tennis

01) Reading data

02) Implementation

```
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. log 2 (fraction)

    return entropy
```

Average Information

$$I(\text{Attribute}) = \Sigma \frac{p_i + n_i}{p+n} \text{ Entropy (Attribute)}$$

Information gain Gain

$$\text{Gain} = \text{Entropy}(s) - I(\text{Attribute})$$

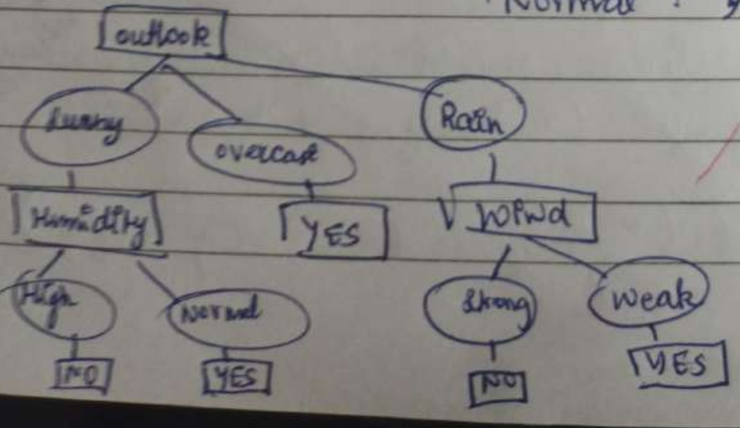
Build Decision Tree

```
def build Tree (df, tree = None) :
        target = df. keys ()[-1]
        node = find winner(df)
        all value = np. unique (df[node])
      if tree is None:
                tree = {}
                tree [node] = {}
    for value in att value:
          subtable = get_subtable(df, node, value)
      cvalue, counts = np. unique [subtable[target],
                         return_counts = True)
      if len(counts) = = 1:
                tree [node][value] = cvalue [0]
      else:
          tree [node][value] = build Tree(subtable)

    return tree

Tree = build Tree (df)
```

output :- { outlook : { 'overcast' : 'yes',
                'Rain' : { 'wind' : { 'strong' : 'NO',
                       'week' : 'yes' }},
                'sunny' : { 'humidity' : { High' : 'NO',
                       'Normal' : 'yes' }}}}



Date
Page

9/5/24

## Logistic Regression.

```
import pandas as pd
from matplotlib import pyplot as plt

file = " insurance.csv"
df = pd.read_csv(file)

from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(x_train, y_train)
print (x_test)

y_predicted = model.predict (x_test)
print (y_predicted)
```

output :

```
Age : 22
probability : 0.1059
```

N
9/5/24

## KNN

```
import pandas as pd
from sklearn.datasets import load_iris
iris = load_iris()

df = pd.Dataframe (iris.data, columns=iris.feature_names)
df['target'] = iris.target
df['flower_name'] = df.target.apply (lambda x : iris.
                                        target_names[x])
```

```
from sklearn.model_selection import train_test_split
x = df.drop(['target', 'flower_name'], axis='columns')
y = df.target

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


from sklearn.neighbors import KNeighbourClassifier
knn = KNeighborsClassifier (n_neighbors=10)
knn.fit(x_train, y_train)

knn.score (x_test, y_test)
   // 0.9666

knn.predict ([[4.8, 3.0, 1.5, 0.3]])

output :- //  0  (serosa)
```

9/5/24

## Lab :- 07

## K Means Implementation

```
import pandas as pd
data = pd. read csv (" Iris. csv ")
 data. head()

import numpy as np
 import seaborn as sns
  import matplotlib . pyplot as plt
from sklearn . datasets import load_iris
from sklearn . cluster import KMeans.

x, y = load_iris (return_x_y = True)

K means = KMeans (n_cluster = 3, random-state =
                                              2)
KMeans. fit (x)

pred = kmeans. fit_predict(x)
 pred
```

SVM

```
from sklearn.datasets import load_breast_cancer
import matplotlib.pyplot as plt
from sklearn.inspect import DecisionBoundaryDisp
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()
```

Date
Page

# PCA

```
import pandas as pd
import numpy as np
import matplotlib . pyplot as plt
% matplotlib inline
from sklearn .decomposition import PCA
from sklearn . preprocessing import Standardscale

from sklearn. datasets import load_breast_cance
 data = loa_breast_cancer()
 data . keys()

print ( data ['feature_names'])

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

scaling = standard Scalar()
 scaling . fit (df1)
scaled_data = scaling. transform ( df1)
 principal = PCA ( n_components = 3)
  pricipal .fit (scaled_data)
  X = principal . transform (scaled_data)
```

30/6/24

30/5/24

Lab :- 08

→ Random Forest Ensemble method & Ada Boost

```
from sklearn.datasets import make_moons
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.tree import Desicion Tree Classifier
from sklearn.ensemble import RandomForest Clasifier,
     Ada Boost Classifier.


x, y = make moons(n_samples = 10000, noise = '5,
                        random state =0)


x train, x_test, y_train, y_test = train_test_split
                     (x, y, test_size =0.2,
                             randomstate =42)


clf = Random Forest Clasifier (n_estimators =100,
        max_features = "auto", random state=0)
clf.fit (x train, y train)

y_pred = clf.predict (x_test)

accuracy_score (y_test, y_pred)
```

Output :- 0.7965

30/5/24

Adaboost :-

```
Clf. = AdaBoost Classifier (n_estimator =100)
 clf. fit (x_train, y_train)
  y pred = clf. predict (x_test)

 accuracy_score (y, test, y_pred)
```

//output :    0.8333

Date 21/3/24
Page

Import and export pandas library functions

```
import pandas as pd

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

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

data.columns = col_names
```

→ output :-

```
    sepal_length_in_cm  sepal_width_in_cm  petal length  petal width
                                                              Class
0
0        4.9                  3.0              1.4          0.2 Iris
                                                               setosa

1        4.7                  3.2              1.3          0.2 Iris-setosa
```

21/3/24

Pandas
lessons

1. Creating, Reading & Writing
2. Indexing Selecting & Assigning
3. Summary Functions & Maps
4. Grouping & Sorting
5. Data types & Missing Values
6. Renaming & Combining

Completed the Kaggle Pandas Certificate