21/03/24

Lab - 1

Q. Write a Python program to import and export csv files using Pandas.

A)

```
import pandas as pd

df = pd.read_csv('./Iris.csv')
df.head()
```

OUT:

Id	SepalLengthCm	SepalWidthCm	PetalLengthCm
1	5.1	3.5	1.4

PetalWidthCm	Species
0.2	Iris-setosa

```
url = 'https://....'
col_names = ['Length in cm', 'width in cm',
             'petal Length in cm', 'petal width in cm',
             'class']

data = pd.read_csv(url, names = col_names)
data.head()
```

OUT:

Sepal Length in cm	Sepal width in cm	petal Length in cm
5.1	3.5	1.4

petal width in cm	class
0.2	Iris-setosa

```
data.to_csv('NewIris.csv')        //export
```

24/3/24

Lab -2

Housing Data

Main steps involved are:

1) Defining the problem
2) Import the data
3) Cleaning of data ; analysing of data
4) Selecting an ML model
5) Training the Model
6) Evaluating the Model's performance
7) Deploying the model.

4)

The data has been imported via the url.

3) Cleaning and Description

To describe the dataset, we used the
- describe and .head functions
- info method can be used to get info on the dataset

The dataset has 10 attributes. and 20,640 instances

The null values in each column can be found out using isnull() method to get count of NAN values.

The data types per column is also visible.

Visualisation:

By making use of matplotlib library histograms on each column can be drawn to

display the continuous data.

For example

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

This makes use of hist function from seaborn library.

4) Creating Test Data

```
shuffled_indices = np.random.permutation(len(data))
test_size = int( len(data) * test_ratio)
test_indices = shuffled_indices[ : test_size]
train_indices = shuffled_indices[ test_size :]

return data.iloc[train_indices], data.iloc[test_indices]
```

This divides the data set into test and train sets with 20% of data in test dataset. The indices are selected randomly.

Sp:1
20/2/24

Lab - 3

* Describe and visualise the data to gain insights.

```
from zlib import crc32

def test_set_check(identifier, test_ration=.2):
    total size = 2**32
    hex_repn = crc32(np.int64(identifier) & 0xffffff
    im_test = hex_repn < (test_ration * total_size)
    return im_test


def split_train_test_by_id(data, test_ration, id):
    ids = data[id]
    im_test_set = id.apply(lambda id: test_set_check(
    id_, test_ratio))
    return data.loc[~im_test_set], data.loc[im_test_set]


train_set, test_set = split_tran_test_by_id(data=
housing_with_id, test_ration=0.2, id="index")
train_set.shape, test_set.shape

((16512,11), (4128,11))
```

1) housing = stat_train_set.copy();
housing.shape

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

3) housing[['population', 'median_house_value']].corr()

	population	median_house_value
population	1.000	-0.026882
median_house_value	-0.026882	1.0000

4) housing.describe()

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

The descriptive statics of dataset are presented including mean, standard deviation and percentiles of numerical attributes.

. Select & Train Model

from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(x= housing_prepared, y= housing.labels)

from sklearn.metrics import mean_squared_error

housing_predictive = lin_reg.predict(housing_prepared)

* Tune the model

from sklearn.model_selection import grid_search
forest_reg = RandomForestRegression()
grid_search.fit(x= housing, y= housing_labels)
grid search CV
estimate: RandomForestRegression.

18/04/25

classmate
Date
Page

Week - 4

Linear Regression

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

def estimate_self (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([1,3,2,5,7,8,9,10,12])
    b = estimate_coeff(x,y)
    print("Estimate coeff: \n b_0 = {} \n b_1 = {}"
          format(b))
```

Output:

(b_0, b_1) = (1.2, 363, ... 1.16969...)

with inbuilt

```
from sklearn.model_selection import columnTransform
from sklearn.compose import columnTransformer.
from sklearn.preprocessing import oneHotEncoder
from sklearn.linear_model import LinearRegression
```

The above can work for simple linear as well as multiple linear regression

On multiple linear regression we predict dependent variable based on multiple independent instead of single one.

20/04/2024

Week - 5

```
import numpy as np
import pandas as pd

eps = np.finfo(float).eps
from numpy import log2 as log

import os

for dirname, _, filenames in os.walk('/kaggle/input'):

for filename in filenames:
    print(os.path.join(dir, file))

df = pd.read_csv('path')
df = df.drop('day', axis=1)
df.head()

print(f"Rows: {df.shape[0]}, columns: {df.shape[1]}")
print(df.columns)
df.info()
df.describe()
```

Handling missing data

$$\text{Entropy} = \frac{-p}{p+n}\log\left(\frac{p}{p+n}\right) - \frac{n}{p+n}\log\left(\frac{n}{p+n}\right)$$

i) we compute entropy for dataset
ii) Take average info entropy for current attribute
iii) calculate gain
iv) pick highest gain attribute
v) Repeat until we get desired tree

Average Information :

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

Information gain

Gain = Entropy(S) - I(Attribute)

Building decision tree

Gain = Entropy(S) - I(Attribute)

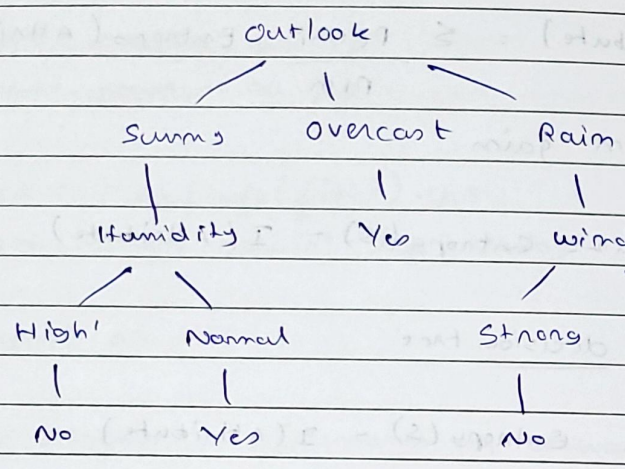
Building Decision Tree

```
def buildTree(df, tree=None):
    target = df.keys()[-1]
    if tree is None:
        tree = {}
        tree[node] = {}

    for value in attribute's attr value:
        subtable = get_subtable(df, node_value)
        if len(counts) == 1:
            tree[node][value] = cl value[0]
        else
            tree[node][value] = buildTree(subtable)
    return tree

tree = tree = buildTree(df)
print.pprint(tree)
```

classmate
Date
Page

04/05/24

Output



Code Output

```
{'outlook': {'overcast': 'Yes', 'Rain': {'wind':
                {'strong': No, 'weak': 'Yes'}},
            'Sunny': {'Humidity': {'High': 'No',
                'Normal': 'Yes'}
}}}
```

04/05/24

classmate
Date
Page

Lab 6

Logistic Regression and KNN

Dataset used: Social_Network_ads

```
from sklearn.linear_model import LogisticRegression

le = LabelEncoder()
df.net['Gender'] = le.fit.transform(df_net['Transform'])

plt.scatter(df.net['Age'], df_net['Estimated Salary'])

x = df.net.iloc[:, :-1].values
y = df_net.iloc[:, -1].values

xtrain, xtest, ytrain, ytest = train_test_split
(x, y, size=0.25, random_state)

sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)

classifier = LogisticRegression(random_state=0)
classifier.fit(x_train, y_train)

y_pred = classifier.predict(x_test)
accuracy_score(y_test, y_pred)
=> 0.83

print(f"F1 score: {f1_score(y_test, y_pred)}")
=> F1score: 0.78

CF_matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(cf_matrix)
```

For KNN fit feature scaling procedure in name

```
from sklearn.neighbours import KNeighbourClassifier as KNN

classifier = KNN(n_neighbours = 5, metric = 'minkowshi', p=2)
classifier.fit(x_train, y_train)
y_pred = classifier.predict(x_test)

accuracy_score(y_test, y_pred)
=> 0.88

print("F1 score: %f F1_score(y_test, y_pred)")
=> F1 score = 0.86

cf_Matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(cf_matrix)
```

For sample data

Logistic Regression, age = 45, salary = 97000
predicted result = 1 (purchased ads)

KNN, Age = 30, salary = 87000
predicted result = 0 (not purchased)

correlation matrix

for LR:

y_pred \ y_true		
0	52	6
1	11	31

for KNN

y_pred \ y_true		
0	50	8
1	4	38

0,0 -> True Negative

0,1 -> False Negative

1,0 -> False Negative

1,1 -> True Positive

23/05/24

Lab 7

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[:, :7]
  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",
  map = plt.cm.Spectral,
  alpha = 0.8,
  xlabel = cancer.feature_name[0];
  ylabel = cancer.feature_name[1];
)
plt.scatter( X[:0] , X[:1] , cy=y , s=20, edgecolor
plt.show()
```

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)

principal = PCA ( n_component = 3)
principal.fit (scalar.data)

x = principal.transform(scaled_data)

plt.figure( figsize = (10,10))
plt.scatter( x[:,0], x[:,1], c =data['target'], cmap='plasma')

plt. xlabel( "PC1')
plt. ylabel ('PC2')
```

k_Means

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

x,y = load_iris (return_x_y = True)
kmeans = kmeans( n_cluster, 3, random_state =2)
kmeans.fit(x)
pred = kmeans.fit_predict(x)
```

30/05/24

Week 8

Ensemble Random Forest

```
from sklearn.datasets import make_moons
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RF, ABC, GBC

x, y = make_moons (n_samples = 10000, noise = .5,
random_state = 0)

x_train, Y_train, x_test, y_test = train_test_split (x, y,
test_size = 02, random_state = 42)

clf = DecisionTree Classifier()
clf.fit ( x_train, y_train)
y_pred = clf.predict (x_test)
accuracy_score (y_test, y_pred)

=> 0.753

clf = Random Forest Classifier ( 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)

=> 0.765

clf = AdaBoost Classifier ( n_estimators = 100)
clf.fit(x_train, y_train)
y_pred = clf.predict (x_test)
accuracy_score (y_test, y_pred)
=> 0.833
```

Sat 30/5/24