11/3/24 — 14-3-24

~~Hotel Management System~~

~~1. End Users (e.g.~~

~~Expour~~

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

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

Output :-

Returns first five rows of the dataset.

* Reading data from URL

```
url = "https://archive.ics.uci.edu/ml/machine-learning
      -databases/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_[illegible]
iris_data.head( )
```

Output:

| | sepal_length_in_cm | sepal_width_in_cm |
|---|---|---|
| 0 | 5.1 | 3.5 |
| 1 | 4.9 | 3 |
| 2 | 4.7 | 3.2 |
| 3 | 4.6 | 3.1 |
| 4 | 5 | 3.6 |

| pedal_length_in_cm | pedal_width_in_cm |
|---|---|
| 1.4 | 0.2 |
| 1.4 | 0.2 |
| 1.3 | 0.2 |
| 1.6 | 0.2 |
| 1.4 | 0.2 |

Exporting dataframe to a CSV file

iris_data.to_csv("cleaned_iris_data.csv")

Output:
iris data dataframe is exported to a CSV file name cleaned_iris_data

# Kaggle

Creating Data:

DataFrame

Eg: pd.DataFrame({'Yes': [50, 21],
'No': [131, 2]})

| | Yes | No |
|---|---|---|
| 0 | 50 | 131 |
| 1 | 21 | 2 |

Series:

Eg:
pd.Series([1, 2, 3, 4, 5])

| | |
|---|---|
| 0 | 1 |
| 1 | 2 |
| 2 | 3 |
| 3 | 4 |
| 4 | 5 |

Reading DataFiles

Eg:- data = pd.read_csv("college data.csv")

data.head( ) returns first five data values of dataset.

28-03-24

Ad 3/24

End-to-End project Hands on Machine Learning

Step 1 :-

Get the Data

data.head( )

- gives first five data entries

data.info( )

output :

| | column | Non-null |
|---|---|---|
| 0 | longitude | 20640 |
| 1 | latitude | 20640 |
| 2 | housing_age | 20640 |
| 3 | total_rooms | 20640 |
| 4 | total_bedrooms | 20640 |
| 5 | population | 20640 |
| 6 | households | 20640 |
| 7 | median_income | 20640 |
| 8 | median_house_income | 20640 |
| 9 | ocean_proximity | 20640 |

```
data['ocean_proximity'].value_counts()

data.describe()

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

Step 2:

Discover & Visualize the Data to gain insights

```
data.plot(kind='scatter', x='longitude',
          y='latitude')
plt.show
```

housing

```
data.plot(kind='scatter', x='longitude', y='latitude',
          alpha=0.1)
plt.show()
```

housing.

```
data[['population', 'median_house_value']].corr()

corr_matrix = housing.corr()
```

corr_matrix['median_house_value'].sort_values
(ascending = False)

data.plot(kind = 'scatter', x = 'median_income',
y = 'median_house_value', figsize = (12
alpha = 0.1)
plt.show()

Step 3:- Prepare the data for Machine Learning Algorithms.

housing_num = data.drop("ocean_proximity",
axis = 1)

housing_cat = housing[['ocean_proximity']]
housing_cat.head()
imputer.fit(housing_num)
~~housing_~~
for train_index, test_index in split.split(
X = housing, y = housing['income_cat']):
strat_train_set = housing.loc[train_index]
strat_test_set = housing.loc[test_index]

housing = strat_train_set.copy().

5. Select and train model

```
from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(X = housing_prepared , y = housing_labels)
some_data = housing.iloc[:5]
print("Predictions :", lin_reg.predict(some_data))

from sklearn.metrics import mean_squared_error
housing_predictions = lin_reg.predict(housing_prepared)
lin_mse = mean_squared_error(housing_labels,
                              housing_predictions)
```

6. Fine-tune your model

```
from sklearn import RandomForestRegressor
from sklearn.model_selection import GridSearch
forest_reg = RandomForestRegressor()
grid_search.fit(X = housing_prepared,
                y = housing_labels)

grid_search.best_params
grid_search.best_estimator
```

12

# Simple Linear Regression

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

df_sal = pd.read_csv('Salary_data.csv')
df_sal.head()
plt.title("Salary Plot")
sns.distplot(df_sal['Salary'])
plt.show()

plt.scatter(df_sal['Years Experience'],
df_sal['Salary'], color = 'lightcoral')
plt.title('Salary vs Experience')
plt.show()
```

## Split Data

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

split into Train and test sets

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size = 0.2, random_state = 0 )

Train Model

regressor = LinearRegression ( )
regressor.fit (X_train, y_train)

y_pred_test = regressor.predict( X_test)
y_pred_train = regressor.predict( X_train)

- Visual predictions

plt.scatter (X_train, y_train, color='lightcoral')
plt.plot (X_train, ~~y_train~~ y_pred_train )
plt.show

Coefficient and Intercept

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

Coefficient : [[9312.57812]]
Intercept : [26780.0991]

# Multiple Linear Regression

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

df_start = pd.read_csv('startups.csv')
df_start.head()
df.describe()
```

Distribution

```
plt.title('Profit Distribution Plot')
sns.distplot(df_start['Profit']
plt.show()
```

Relationship between Profit and R&D spend

```
plt.scatter(df_start['R&D spend'], df_start['Pro
plt.show()
```

Split Data

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

split into train / test data

```
X_train, ~~X_train~~ X_test, y_train, y_test =
train_test_split(X, y, test_size = 0.2,
  random_state = 0)
```

Train Model

```
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

Predict Results

```
~~y_te~~ y_pred = regressor.predict(X_test)
```

compare prediction

```
np.set_printoptions(precision=2)
result = np.concatenate((y_pred.reshape(len(y_pred),
         1), y_test.reshape(len(y_test)), 1))
print(result)

array([[103015.2, 103282.38],
       [1032582.26, 144259.4],
       [132447.74, 146121.95],
      ])
```

18/04

Decision Tree using Entrophy

A 18-4-24

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt

iris_data = pd.read_csv('Iris.csv')
iris_data.head()

iris_data.drop('Id', inplace = True, axis = 1)
X = iris_data.drop('Species', axis = 1)
y = iris_data['Species']

X_train, X_test, y_train, y_test = train_test_split
        (X, y, test_size = 0.4, random_state = 42)
dt_classifier = DecisionTreeClassifier(criterion = 'entrophy'
                random_state = 42)

dt_classifier.fit(X_train, y_train)
```

```python
plt.figure(figsize(12, 8))
plot_tree(dt_classifier, feature_name = X.colu
          class_names = dt_classifier.classes_
          filled = True)
plt.show()

y_pred = dt_classifier.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy : {accuracy}")
```

Accuracy = 0.98333

PetalLengthCm <= 2.45
entropy = 1.581
samples = 90
value = [27, 31, 32]
class = Iris - virginica

entropy = 0.0
sample = 27
value = [27, 0, 0]
class = Iris - setosa

PetalWidthCm <= 1.
entropy = 1.0
samples = 63
value = [0, 31, 32]
class = Iris - virginica

25/4/24

## Build Logistic Regression Model for a given dataset

```
import pandas as pd
from matplotlib import pyplot as plt
%matplotlib inline
df = pd.read_csv("insurance_data.csv")
df.head()
plt.scatter(df.age, df.bought_insurance, marker='+',
            color='red')

from sklearn.model_selection import train_test_
    split
X_train, X_test, y_train, y_test = train_test_
split(df[['age']], df.bought_insurance,
    train_size=0.8)

model = LogisticRegression()
model.fit(X_train, y_train)

y_predicted = model.predict(X_test)

print(y_predicted)
[1 1 0 1 1 0]
```

print(X_test)

| | age |
|---|---|
| 9 | 61 |
| 23 | 45 |
| 21 | 26 |
| 4 | 46 |
| 7 | 60 |
| 0 | 22 |

print("Model Accuracy", model.score(X_test, y

Model Accuracy 0.8333

~~print(y_pred~~
print("Coefficient", model.coef_)
print("Intercept", model.intercept_)

Coefficient [[0.14999]]
Intercept [[-5.6081]]

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

def prediction_function(age):
```

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

age = 35
prediction_function(age)

0.485004

age = 43
prediction_function(age)

0.568565
```

12/03/2024 # KNN and SVM

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.neighbors import KNeighborsClassifier
~~from sklearn.tree imp~~
import matplotlib.pyplot as plt

iris_data = pd.read_csv('Iris.csv')
iris_data.head()

label_mapping = {'Iris-setosa': 1, 'Iris-versicolor'
                 'Iris-virginica': 3}
iris_data['Species'] = iris_data['Species'].map(
        label_mapping)

plt.scatter(iris_data['SepalLengthCm'], iris_data
        ['SepalWidthCm'], c=iris_data['Species']
        cmap='viridis')
plt.show()

iris_data.drop('Id', inplace=True, axis=1)
X = iris_data.drop('Species', axis=1)
y = iris_data['Species']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
                test_size = 0.4, random_state = 42)

# KNN
knn = KNeighboursClassifier(n_neighbour = 5)
knn.fit(X_train, y_train)

y_pred = knn.predict(X_test)

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

⟹ Accuracy: 0.98333

```
# SVM
svm_classifier = SVC()
svm_classifier.fit(X_train, y_train)
y_pred = svm_classifier.predict(X_test)

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

⟹ Accuracy = 0.93333

D 23-5-24

23-05-24 Random Forest Regression & ADABoost

```
import pandas as pd
from sklearn.model_selection import train
from sklearn.ensemble import RandomForestClass
~~from sklearn.~~
from sklearn.model selection import accuracy

iris_data = pd.read_csv('Iris.csv')
iris data.head( )
iris data.info( )

iris_data = pd.read_csv('Iris.csv')
iris data.head( )

iris data.drop('Id', inplace='True', axis=1)
X = iris data.drop('Species', axis=1)
y = iris data['Species']
iris data.info( )

X_train, X_test, y_train, y_test = train
  (X, y, test_size = 0.4, random_state

a) rf classifier = RandomForestClassifier(n_estimators
                                   random_state = 42
```

~~rf~~ classifier.fit(X_train, y_train)

y_pred = rf classifier.predict(X_test)

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

Accuracy: 0.9833

b) mylogregmodel = LogisticRegression( )
~~d~~ adabc = ~~ab~~ AdaBoostClassifier(n_estimators = 150,
base_estimators = mylogregmodel, learning_rate = 1)

model = adabc.fit(X_train, y_train)

y_pred = model.predict(X_test)

metrics.accuracy_score(y_test, y_pred)

0.98333

03-05-2024

# ANN

```
import numpy as np
X = np.array(([2,9], [1,5], [3,6]), dtype=f
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_n
                hiddenlayer_neurons))
bh = np.random.uniform(size = (1, hiddenlayer
                neurons))
wout = np.random.uniform(size = (hiddenlayer_n
                outputlayer_neurons)
bout = np.random.uniform(size = (1, output_n

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)
    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_output) * lr
    wh += X.T.dot(d_hiddenlayer) * lr

print("Input:\n" + str(X))
print("Actual Output:\n" + str(y))
print("Predicted Output:\n" + output)
```

Input:

[[0.6666667, 1.        ],
[0.3333          0.5555  ].
[1.              0.6666  ]]

Actual Output:

[[0.92]
[0.86]
[0.89]]

Predicted Output:

[[0.807111]
[0.790364]
[0.8011414]]

30-05-24 K-Means Clustering 30-5-24

```
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.Petal_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=4
plt.title('K-means clustering')
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
```

-06-24 PCA (Principal Component Analysis)

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

from sklearn.datasets import load_breast_cancer
cancer = load_breast_cancer()
cancer.keys()
print(cancer['DESCR']

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

from sklearn.preprocessing import StandardScalar
scalar = StandardScalar
scalar.fit(df)

scaled_data = scalar.transform(df)

from sklearn.decomposition import PCA

pca = PCA(n_components=2)
pca.fit(scaled_data)
```

```
plt.figure(figsize(8,6))
plt.scatter(x_pca[:, 0], x_pca[:, 1],
            c= cancer['target'], cmap='plas
plt.show()
```