Project title: Airlines Data Analytics for Avaition Industry

PNT2022TMID37380

In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from pandas.api.types import is_numeric_dtype
sns.set()
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
sns.set_style("darkgrid")
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor

from sklearn import  metrics
%matplotlib inline
```

2 LOAD THE DATASET

In [ ]:
```
df=pd.read_csv("/content/abalone.csv")
```

In [ ]:
```
df['age'] = df['Rings']+1.5
df = df.drop('Rings', axis = 1)
```

3 i) UNIVARIATE ANALYSIS

In [ ]:
```
df.hist(figsize=(20,10), grid=False, layout=(2, 4), bins = 30)
```

Out[ ]:
```
array([[,
        ,
        ,
        ],
       [,
        ,
        ,
        ]],
      dtype=object)
```

In [ ]:
```
df.groupby('Sex')[['Length', 'Diameter', 'Height', 'Whole weight', 'Shucked weight',
       'Viscera weight', 'Shell weight', 'age']].mean().sort_values('age')
```

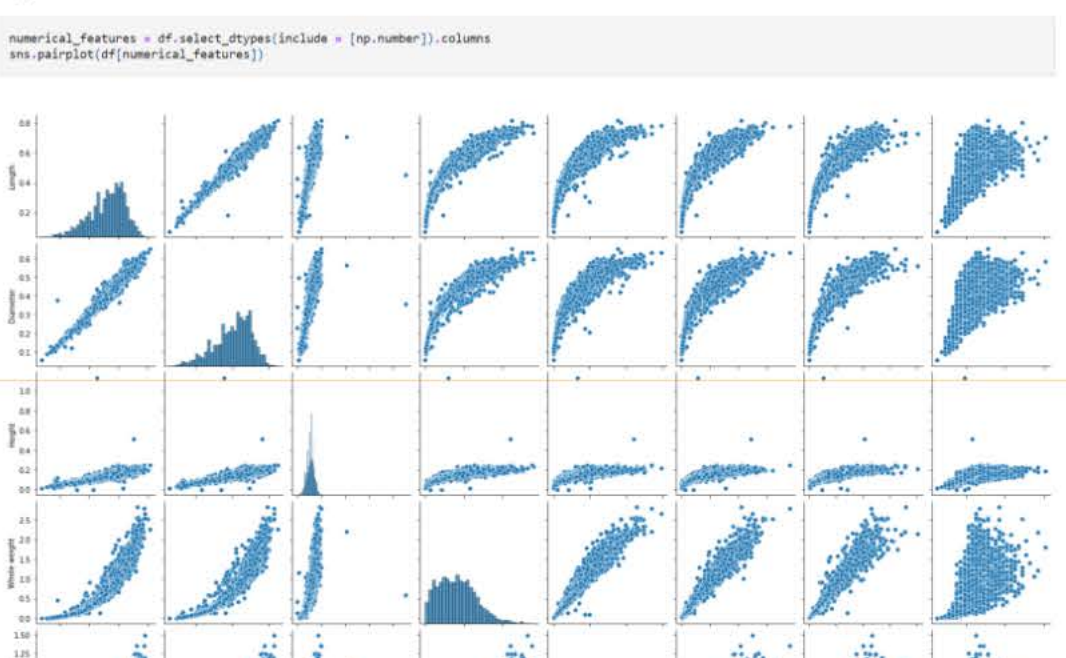
Out[ ]:

Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	age
I	0.427746	0.326494	0.107996	0.431363	0.191035	0.092010	0.128182	9.390462
M	0.561391	0.439287	0.151381	0.991459	0.432946	0.215545	0.281969	12.205497
F	0.579093	0.454732	0.158011	1.046532	0.446188	0.230689	0.302010	12.629304

3 ii) BIVARIATE ANALYSIS & MULTIVARIATE ANALYSIS

In [ ]:
```
numerical_features = df.select_dtypes(include = [np.number]).columns
sns.pairplot(df[numerical_features])
```

Out[ ]:



4 Perform descriptive statistics on the dataset

In [ ]:
```
df.describe()
```

Out[ ]:

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	age
count	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000
mean	0.523992	0.407881	0.139516	0.828742	0.359367	0.180594	0.238831	11.433684
std	0.120093	0.099240	0.041827	0.490389	0.221963	0.109614	0.139203	3.224169
min	0.075000	0.055000	0.000000	0.002000	0.001000	0.000500	0.001500	2.500000
25%	0.450000	0.350000	0.115000	0.441500	0.186000	0.093500	0.130000	9.500000
50%	0.545000	0.425000	0.140000	0.799500	0.336000	0.171000	0.234000	10.500000
75%	0.615000	0.480000	0.165000	1.153000	0.502000	0.253000	0.329000	12.500000
max	0.815000	0.650000	1.130000	2.825500	1.488000	0.760000	1.005000	30.500000

5. Check for Missing values

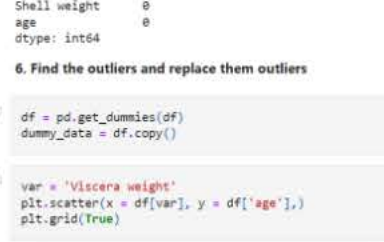
In [ ]:
```
df.isnull().sum()
```

Out[ ]:
```
Sex               0
Length            0
Diameter          0
Height            0
Whole weight      0
Shucked weight    0
Viscera weight    0
Shell weight      0
age               0
dtype: int64
```

6. Find the outliers and replace them outliers

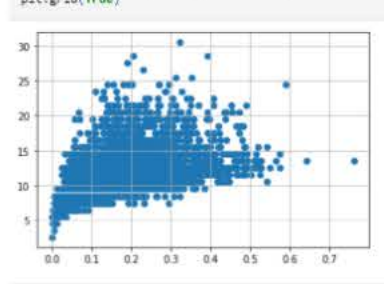
In [ ]:
```
df = pd.get_dummies(df)
dummy_data = df.copy()
```

In [ ]:
```
var = 'Viscera weight'
plt.scatter(x = df[var], y = df['age'],)
plt.grid(True)
```



In [ ]:
```
# outliers removal
df.drop(df[(df['Viscera weight']> 0.5) & (df['age'] < 20)].index, inplace=True)
df.drop(df[(df['Viscera weight']<0.5) & (df['age'] > 25)].index, inplace=True)
```

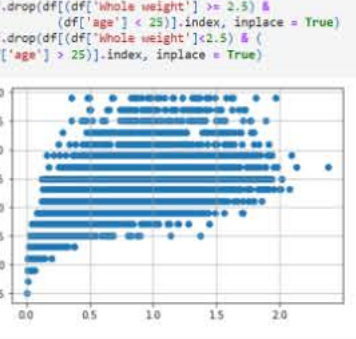
In [ ]:
```
var = 'Shell weight'
plt.scatter(x = df[var], y = df['age'],)
plt.grid(True)
#Outliers removal
df.drop(df[(df['Shell weight']> 0.6) & (df['age'] < 25)].index, inplace=True)
df.drop(df[(df['Shell weight']<0.8) & (df['age'] > 25)].index, inplace=True)
```



In [ ]:
```
var = 'Shucked weight'
plt.scatter(x = df[var], y = df['age'],)
plt.grid(True)
#Outlier removal
df.drop(df[(df['Shucked weight']>= 1) & (df['age'] < 20)].index, inplace=True)
df.drop(df[(df['Shucked weight']<1) & (df['age'] > 20)].index, inplace=True)
```

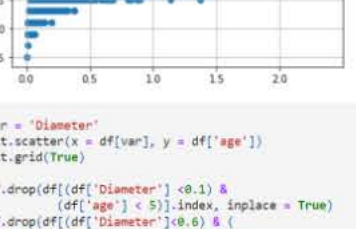
In [ ]:
```
var = 'Whole weight'
plt.scatter(x = df[var], y = df['age'])
plt.grid(True)

df.drop(df[(df['Whole weight'] >= 2.5) &
          (df['age'] < 25)].index, inplace = True)
df.drop(df[(df['Whole weight']<2.5) & (
df['age'] > 25)].index, inplace = True)
```

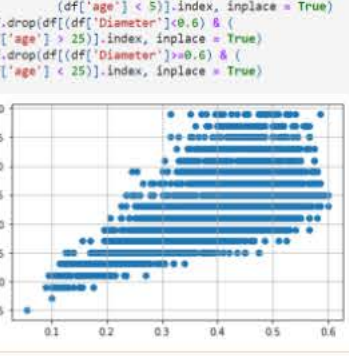


In [ ]:
```
var = 'Diameter'
plt.scatter(x = df[var], y = df['age'])
plt.grid(True)

df.drop(df[(df['Diameter'] <0.1) &
          (df['age'] < 5)].index, inplace = True)
df.drop(df[(df['Diameter']<0.6) & (
df['age'] > 25)].index, inplace = True)
df.drop(df[(df['Diameter']>=0.6) & (
df['age'] < 25)].index, inplace = True)
```

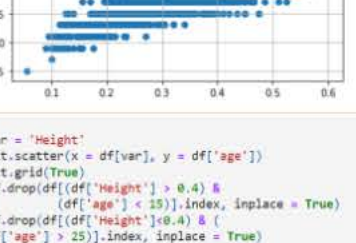


In [ ]:
```
var = 'Height'
plt.scatter(x = df[var], y = df['age'])
plt.grid(True)
df.drop(df[(df['Height'] > 0.4) &
          (df['age'] < 15)].index, inplace = True)
df.drop(df[(df['Height']<0.4) & (
df['age'] > 25)].index, inplace = True)
```



In [ ]:
```
var = 'Length'
plt.scatter(x = df[var], y = df['age'])
plt.grid(True)

df.drop(df[(df['Length'] <0.1) &
          (df['age'] < 5)].index, inplace = True)
df.drop(df[(df['Length']<0.8) & (
df['age'] > 25)].index, inplace = True)
df.drop(df[(df['Length']>=0.8) & (
df['age'] < 25)].index, inplace = True)
```



7. Check for Categorical columns and perform encoding.

In [ ]:
```
numerical_features = df.select_dtypes(include = [np.number]).columns
categorical_features = df.select_dtypes(include = [np.object]).columns
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: DeprecationWarning: `np.object` is a deprecated alias for the builtin `object`. To s
ilence this warning, use `object` by itself. Doing this will not modify any behavior and is safe.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
```

In [ ]:
```
numerical_features
```

Out[ ]:
```
Index(['Length', 'Diameter', 'Height', 'Whole weight', 'Shucked weight',
       'Viscera weight', 'Shell weight', 'age', 'Sex_F', 'Sex_I', 'Sex_M'],
      dtype='object')
```

In [ ]:
```
categorical_features
```

Out[ ]:
```
Index([], dtype='object')
```

ENCODING

In [ ]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
print(df.Length.value_counts())
```

```
0.575    93
0.625    91
0.580    89
0.550    89
0.620    83
         ..
0.220     2
0.150     1
0.755     1
0.135     1
0.760     1
Name: Length, Length: 126, dtype: int64
```

8. Split the data

into dependent and independent variables

In [ ]:
```
x=df.iloc[:,:5]
x
```

Out[ ]:

	Length	Diameter	Height	Whole weight	Shucked weight
0	0.455	0.365	0.095	0.5140	0.2245
1	0.350	0.265	0.090	0.2255	0.0995
2	0.530	0.420	0.135	0.6770	0.2565
3	0.440	0.365	0.125	0.5160	0.2155
4	0.330	0.255	0.080	0.2050	0.0895
...	...	...	...	...	...
4172	0.565	0.450	0.165	0.8870	0.3700
4173	0.590	0.440	0.135	0.9660	0.4390
4174	0.600	0.475	0.205	1.1760	0.5255
4175	0.625	0.485	0.150	1.0945	0.5310
4176	0.710	0.555	0.195	1.9485	0.9455

3995 rows × 5 columns

In [ ]:
```
y=df.iloc[:,5:]
y
```

Out[ ]:

	Viscera weight	Shell weight	age	Sex_F	Sex_I	Sex_M
0	0.1010	0.1500	16.5	0	0	1
1	0.0485	0.0700	8.5	0	0	1
2	0.1415	0.2100	10.5	1	0	0
3	0.1140	0.1550	11.5	0	0	1
4	0.0395	0.0550	8.5	0	1	0
...	...	...	...	...	...	...
4172	0.2390	0.2490	12.5	1	0	0
4173	0.2145	0.2605	11.5	0	0	1
4174	0.2875	0.3080	10.5	0	0	1
4175	0.2610	0.2960	11.5	1	0	0
4176	0.3765	0.4950	13.5	0	0	1

3995 rows × 6 columns

9. Scale the independent

variables

In [ ]:
```
#Scaling the Independent Variables
print ("\n ORIGINAL VALUES: \n\n", x,y)
```

```
 ORIGINAL VALUES:

       Length  Diameter  Height  Whole weight  Shucked weight
0      0.455     0.365   0.095        0.5140          0.2245
1      0.350     0.265   0.090        0.2255          0.0995
2      0.530     0.420   0.135        0.6770          0.2565
3      0.440     0.365   0.125        0.5160          0.2155
4      0.330     0.255   0.080        0.2050          0.0895
...      ...       ...     ...           ...             ...
4172   0.565     0.450   0.165        0.8870          0.3700
4173   0.590     0.440   0.135        0.9660          0.4390
4174   0.600     0.475   0.205        1.1760          0.5255
4175   0.625     0.485   0.150        1.0945          0.5310
4176   0.710     0.555   0.195        1.9485          0.9455

[3995 rows x 5 columns]        Viscera weight  Shell weight   age  Sex_F  Sex_I  Sex_M
0             0.1010        0.1500  16.5      0      0      1
1             0.0485        0.0700   8.5      0      0      1
2             0.1415        0.2100  10.5      1      0      0
3             0.1140        0.1550  11.5      0      0      1
4             0.0395        0.0550   8.5      0      1      0
...              ...           ...   ...    ...    ...    ...
4172          0.2390        0.2490  12.5      1      0      0
4173          0.2145        0.2605  11.5      0      0      1
4174          0.2875        0.3080  10.5      0      0      1
4175          0.2610        0.2960  11.5      1      0      0
4176          0.3765        0.4950  13.5      0      0      1

[3995 rows x 6 columns]
```

In [ ]:
```
from sklearn import preprocessing
min_max_scaler = preprocessing.MinMaxScaler(feature_range =(0, 1))
new_y= min_max_scaler.fit_transform(x,y)
print ("\n VALUES AFTER MIN MAX SCALING: \n\n", new_y)
```

```
 VALUES AFTER MIN MAX SCALING:

 [[0.51587302 0.54545455 0.38       0.21240245 0.22199798]
 [0.34920635 0.34343434 0.36       0.09069816 0.09586276]
 [0.63492063 0.65656566 0.54       0.28116431 0.2542886 ]
 ...
 [0.74603175 0.76767677 0.82       0.49166842 0.52573158]
 [0.78571429 0.78787879 0.6        0.45728749 0.53128153]
 [0.92063492 0.92929293 0.78       0.81754904 0.94954591]]
```

10. Split the data into training and testing

In [ ]:
```
#Split the data into Training and Testing
X = df.drop('age', axis = 1)
y = df['age']
```

In [ ]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.feature_selection import SelectKBest
standardScale = StandardScaler()
standardScale.fit_transform(X)

selectkBest = SelectKBest()
X_new = selectkBest.fit_transform(X, y)

X_train, X_test, y_train, y_test = train_test_split(X_new, y, test_size = 0.25)
X_train
```

Out[ ]:
```
array([[0.505, 0.395, 0.145, ..., 1.   , 0.   , 0.   ],
       [0.3  , 0.22 , 0.065, ..., 0.   , 1.   , 0.   ],
       [0.565, 0.45 , 0.16 , ..., 1.   , 0.   , 0.   ],
       ...,
       [0.33 , 0.255, 0.095, ..., 0.   , 0.   , 1.   ],
       [0.585, 0.46 , 0.145, ..., 0.   , 0.   , 1.   ],
       [0.52 , 0.395, 0.125, ..., 0.   , 1.   , 0.   ]])
```

In [ ]:
```
y_train
```

Out[ ]:
```
2319    16.5
3106     6.5
548     13.5
3968     7.5
1774     8.5
        ...
3353     9.5
2317    14.5
643      8.5
1150    10.5
1294    10.5
Name: age, Length: 2996, dtype: float64
```

11. Build the Model

In [ ]:
```
# Build the model
# Linear Regression
```

In [ ]:
```
from sklearn import linear_model as lm
from sklearn.linear_model import LinearRegression
model=lm.LinearRegression()
results=model.fit(X_train,y_train)
```

In [ ]:
```
accuracy = model.score(X_train, y_train)
print('Accuracy of the model:', accuracy)
```

```
Accuracy of the model: 0.5264583383306936
```

12. Train the Model

In [ ]:
```
#Training the model
lm = LinearRegression()
lm.fit(X_train, y_train)
y_train_pred = lm.predict(X_train)
y_train_pred
```

Out[ ]:
```
array([11.77155364,  7.46499302, 12.15326082, ...,  9.427397  ,
       10.75430984, 10.19548647])
```

In [ ]:
```
X_train
```

Out[ ]:
```
array([[0.505, 0.395, 0.145, ..., 1.   , 0.   , 0.   ],
       [0.3  , 0.22 , 0.065, ..., 0.   , 1.   , 0.   ],
       [0.565, 0.45 , 0.16 , ..., 1.   , 0.   , 0.   ],
       ...,
       [0.33 , 0.255, 0.095, ..., 0.   , 0.   , 1.   ],
       [0.585, 0.46 , 0.145, ..., 0.   , 0.   , 1.   ],
       [0.52 , 0.395, 0.125, ..., 0.   , 1.   , 0.   ]])
```

In [ ]:
```
y_train
```

Out[ ]:
```
2319    16.5
3106     6.5
548     13.5
3968     7.5
1774     8.5
        ...
3353     9.5
2317    14.5
643      8.5
1150    10.5
1294    10.5
Name: age, Length: 2996, dtype: float64
```

In [ ]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error
s = mean_squared_error(y_train, y_train_pred)
print('Mean Squared error of training set :%2f'%s)
```

```
Mean Squared error of training set :3.489912
```

13. Test the Model

In [ ]:
```
y_train_pred = lm.predict(X_train)
y_test_pred = lm.predict(X_test)
```

In [ ]:
```
y_test_pred
```

Out[ ]:
```
array([1.64959689e+01, 9.57924403e+00, 1.10301174e+01, 1.45499784e+01,
       8.62455954e+00, 1.08291515e+01, 1.51660845e+01, 8.88584612e+00,
       1.14015203e+01, 1.32444874e+01, 1.06444475e+01, 8.43992991e+00,
       1.06698405e+01, 1.06305962e+01, 9.60254792e+00, 1.08057883e+01,
       1.15097756e+01, 7.48708326e+00, 1.18479529e+01, 1.10835635e+01,
       1.06601624e+01, 1.40873792e+01, 7.42150480e+00, 1.13218264e+01,
       1.17282441e+01, 1.25579126e+01, 1.35380614e+01, 1.64974465e+01,
       1.28497835e+01, 1.20702397e+01, 1.17673379e+01, 1.18055166e+01,
       1.11441671e+01, 1.03778655e+01, 9.08043138e+00, 1.01999187e+01,
       9.71639924e+00, 1.16798938e+01, 1.18441882e+01, 1.06880535e+01,
       1.23801888e+01, 1.24236437e+01, 1.17491344e+01, 6.21782742e+00,
       7.30698986e+00, 1.20974527e+01, 1.03050316e+01, 1.18446002e+01,
       9.66561958e+00, 1.32346404e+01, 1.25225352e+01, 8.78198540e+00,
       9.62654189e+00, 5.69924377e+00, 1.09582642e+01, 8.02649077e+00,
       1.12516436e+01, 9.30884997e+00, 1.09766734e+01, 1.30403869e+01,
       8.24705330e+00, 9.70632029e+00, 1.17688336e+01, 9.80592912e+00,
       9.76326845e+00, 1.61709206e+01, 9.46905144e+00, 1.44202280e+01,
       9.54799255e+00, 1.10625281e+01, 8.98300712e+00, 1.13198575e+01,
       1.10657615e+01, 1.20093829e+01, 1.14493569e+01, 1.04450662e+01,
       1.18025616e+01, 8.23930950e+00, 1.20854751e+01, 1.01199826e+01,
       1.05878388e+01, 8.73146938e+00, 1.44050294e+01, 1.08502379e+01,
       9.95148639e+00, 1.12779026e+01, 1.07218357e+01, 8.97523541e+00,
       1.22236674e+01, 1.28195094e+01, 1.07264793e+01, 8.60896338e+00,
       1.15199337e+01, 1.13663432e+01, 5.76714364e+00, 8.21495459e+00,
       1.11261326e+01, 1.39506576e+01, 1.31290249e+01, 1.19115562e+01,
       8.54185487e+00, 8.13200766e+00, 1.06258387e+01, 1.40052689e+01,
       1.22285265e+01, 1.10036148e+01, 9.33358923e+00, 1.34924945e+01,
       9.36170450e+00, 7.10174626e+00, 1.31876295e+01, 9.72037342e+00,
       1.34911519e+01, 1.08837382e+01, 9.16303412e+00, 1.09473911e+01,
       8.78692837e+00, 1.16574499e+01, 1.25794010e+01, 9.70239432e+00,
       1.29527078e+01, 1.10011151e+01, 8.93530816e+00, 1.22464379e+01,
       1.04141492e+01, 1.50598552e+01, 9.32473187e+00, 9.37065227e+00,
       1.20862857e+01, 1.22384802e+01, 1.09769579e+01, 1.01130078e+01,
       1.01111469e+01, 8.48294102e+00, 1.18438440e+01, 1.06628483e+01,
       1.14204531e+01, 1.10071589e+01, 9.91428303e+00, 1.38806362e+01,
       6.84213103e+00, 1.22679470e+01, 1.47632037e+01, 8.34971614e+00,
       9.47163593e+00, 1.33869396e+01, 1.18402186e+01, 1.20796670e+01,
       1.22354952e+01, 9.30224750e+00, 1.19902666e+01, 9.22764059e+00,
       9.41452794e+00, 8.99583278e+00, 1.29517659e+01, 1.13194681e+01,
       1.24006218e+01, 1.31733694e+01, 1.12756788e+01, 1.19761367e+01,
       1.43992430e+01, 1.03799245e+01, 1.27457018e+01, 1.11260234e+01,
       9.72365659e+00, 9.59006061e+00, 1.34983564e+01, 1.22172683e+01,
       1.16571953e+01, 8.54065335e+00, 9.04754540e+00, 8.69896649e+00,
       7.24548542e+00, 7.89735132e+00, 1.28427105e+01, 1.03166237e+01,
       1.04700682e+01, 1.40122389e+01, 1.07370858e+01, 1.11034894e+01,
       1.14473567e+01, 9.09682860e+00, 1.55032747e+01, 9.91948075e+00,
       1.25682405e+01, 1.20253405e+01, 1.19244506e+01, 1.06990948e+01,
       1.04703564e+01, 7.66273235e+00, 1.18300285e+01, 9.19968411e+00,
       1.06716688e+01, 1.47918430e+01, 1.10274215e+01, 7.03681665e+00,
       1.10996424e+01, 1.18061350e+01, 1.00660294e+01, 1.44686624e+01,
       1.03055561e+01, 1.04636895e+01, 1.39160853e+01, 9.34330829e+00,
       1.03114811e+01, 8.18694621e+00, 8.67159995e+00, 1.24569750e+01,
       1.34096375e+01, 1.15306979e+01, 9.47770566e+00, 6.30214847e+00,
       1.17083291e+01, 9.46388989e+00, 1.03116381e+01, 9.69619592e+00,
       1.07412536e+01, 1.12359338e+01, 1.27035005e+01, 1.19474798e+01,
       1.15983442e+01, 1.17991746e+01, 1.42483703e+01, 1.40008124e+01,
       1.26042055e+01, 9.96382116e+00, 1.13909193e+01, 1.31830512e+01,
       8.87417521e+00, 1.10818715e+01, 1.41420329e+01, 9.26770944e+00,
       1.09948365e+01, 1.60024077e+01, 9.80133664e+00, 1.34905424e+01,
       7.63789833e+00, 1.06941036e+01, 1.27145520e+01, 1.30726255e+01,
       1.10422393e+01, 1.08789042e+01, 9.74890165e+00, 1.08312717e+01,
       8.69674775e+00, 1.07132505e+01, 8.24507554e+00, 1.34548633e+01,
       1.29566746e+01, 9.41127386e+00, 9.59530695e+00, 7.88769786e+00,
       1.21634570e+01, 1.33799501e+01, 9.82537167e+00, 9.43338786e+00,
       1.19664298e+01, 1.30270114e+01, 1.44841572e+01, 1.17042697e+01,
       9.80787098e+00, 1.23252014e+01, 1.31002910e+01, 1.08888985e+01,
       1.53208892e+01, 1.14307141e+01, 1.18243432e+01, 1.36645765e+01,
       1.00479559e+01, 1.50257187e+01, 8.10141693e+00, 1.13312446e+01,
       9.76060689e+00, 1.26700288e+01, 1.46966117e+01, 1.55482227e+01,
       1.22173603e+01, 1.17959196e+01, 1.02398621e+01, 1.19553217e+01,
       8.74171023e+00, 1.05935040e+01, 8.86481457e+00, 1.52885516e+01,
       1.18894999e+01, 1.34646551e+01, 8.52970965e+00, 9.47665692e+00,
       1.09935354e+01, 7.09452002e+00, 1.27250183e+01, 1.02568303e+01,
       1.48776677e+01, 1.15654722e+01, 1.10565073e+01, 1.02941608e+01,
       9.84863375e+00, 1.16652082e+01, 8.50832100e+00, 8.62335432e+00,
       8.35222272e+00, 8.22929298e+00, 1.34050641e+01, 8.79854614e+00,
       1.24533939e+01, 8.06694299e+00, 9.09164405e+00, 1.03939785e+01,
       1.43779206e+01, 1.03332145e+01, 7.91753943e+00, 1.30473983e+01,
       1.31222850e+01, 1.25767282e+01, 8.23742924e+00, 1.27756370e+01,
       9.64257382e+00, 1.00861152e+01, 9.10645376e+00, 8.84396268e+00,
       1.12143564e+01, 1.23499356e+01, 1.06061954e+01, 9.18289092e+00,
       1.15483240e+01, 9.31907036e+00, 1.38159699e+01, 9.65147066e+00,
       1.24768299e+01, 1.29357825e+01, 1.23460726e+01, 1.05313146e+01,
       1.02336754e+01, 1.16965040e+01, 1.28890137e+01, 1.07289215e+01,
       1.43730047e+01, 1.27632773e+01, 9.78448422e+00, 7.40120646e+00,
       1.16410527e+01, 1.41009401e+01, 8.89080153e+00, 1.27862376e+01,
       1.14731819e+01, 1.15629459e+01, 9.73386399e+00, 9.98639656e+00,
       9.83692720e+00, 8.07419851e+00, 9.81094979e+00, 1.02844760e+01,
       1.10380371e+01, 1.09124326e+01, 7.15334686e+00, 1.35104072e+01,
       1.70950063e+01, 1.11385470e+01, 1.48193601e+01, 1.58746255e+01,
       8.53810490e+00, 1.00690588e+01, 8.15365698e+00, 8.48937354e+00,
       1.04169020e+01, 1.20671657e+01, 1.19495751e+01, 1.29156432e+01,
       1.43828552e+01, 1.17554535e+01, 1.24048489e+01, 1.30409069e+01,
       1.19972302e+01, 1.47368599e+01, 1.17348076e+01, 1.05299883e+01,
       1.16647277e+01, 9.31531854e+00, 1.06490170e+01, 1.24255300e+01,
       1.06250587e+01, 1.01578188e+01, 1.41928794e+01, 9.27627903e+00,
       9.35504468e+00, 1.12407281e+01, 7.73000018e+00, 9.61165565e+00,
       9.99586646e+00, 1.04185834e+01, 1.05517755e+01, 9.68945981e+00,
       1.20094233e+01, 8.99774296e+00, 1.01656736e+01, 1.38477302e+01,
       1.35388028e+01, 1.07878237e+01, 1.05247357e+01, 8.51926197e+00,
       1.03798327e+01, 1.00680998e+01, 1.38161105e+01, 6.34041026e+00,
       9.46329156e+00, 1.49439457e+01, 1.10160412e+01, 1.12830304e+01,
       5.70998904e+00, 1.11515804e+01, 1.19322641e+01, 1.18849466e+01,
       1.11587211e+01, 8.94341001e+00, 1.07493991e+01, 1.10760232e+01,
       1.05919041e+01, 1.42086766e+01, 8.98687096e+00, 1.15242089e+01,
       1.19091874e+01, 1.17232310e+01, 1.19121666e+01, 1.36902867e+01,
       1.16010449e+01, 1.11313985e+01, 9.56689937e+00, 1.21772247e+01,
       1.12999959e+01, 1.01310474e+01, 1.25558297e+01, 1.37074411e+01,
       1.03293776e+01, 1.01979905e+01, 9.25833280e+00, 1.05309556e+01,
       1.06287536e+01, 1.09087580e+01, 1.12849426e+01, 1.11596466e+01,
       1.07819028e+01, 1.20660231e+01, 1.09417859e+01, 1.00960129e+01,
       7.58976822e+00, 9.02083880e+00, 1.28796077e+01, 1.11691642e+01,
       1.05518745e+01, 9.99274453e+00, 1.12421087e+01, 1.27164953e+01,
       1.17922854e+01, 1.20954257e+01, 1.12292136e+01, 1.32644760e+01,
       1.12549362e+01, 7.65980138e+00, 8.51702191e+00, 8.29079763e+00,
       1.26967235e+01, 1.02673253e+01, 9.41900425e+00, 1.25270379e+01,
       1.52399963e+01, 1.04272327e+01, 8.44330874e+00, 1.35226497e+01,
       1.00461154e+01, 1.39596742e+01, 1.12992831e+01, 1.33681925e+01,
       9.56599811e+00, 1.22368345e+01, 1.02573815e+01, 1.17110439e+01,
       1.12717168e+01, 1.10696880e+01, 1.10422327e+01, 9.60012430e+00,
       1.42431788e+01, 1.47820827e+01, 1.38442364e+01, 7.97878652e+00,
       1.01511242e+01, 1.32602471e+01, 8.84969380e+00, 1.50156234e+01,
       1.03586277e+01, 1.09628051e+01, 1.14760034e+01, 1.20003920e+01,
       1.03324701e+01, 1.35179616e+01, 1.17219428e+01, 1.26346858e+01,
       9.17522262e+00, 8.87840173e+00, 7.19978954e+00, 9.73505677e+00,
       1.67149739e+01, 1.24802421e+01, 1.23691842e+01, 1.30477897e+01,
       1.12152016e+01, 1.04725101e+01, 1.01124321e+01, 9.78145739e+00,
       1.03557584e+01, 1.04493486e+01, 1.14498979e+01, 1.19358725e+01,
       7.96554620e+00, 1.23309628e+01, 1.28235842e+01, 1.36242818e+01,
       1.30088166e+01, 9.09050743e+00, 1.18114581e+01, 9.40314646e+00,
       1.36264209e+01, 8.79704216e+00, 1.23598114e+01, 1.05446142e+01,
       1.13255374e+01, 1.01270036e+01, 1.08385226e+01, 1.10812792e+01,
       1.44823221e+01, 1.06296309e+01, 1.10630150e+01, 1.15599883e+01,
       1.56758342e+01, 1.16302430e+01, 1.15011432e+01, 8.86141880e+00,
       9.78953747e+00, 1.21121199e+01, 1.05159404e+01, 7.96304455e+00,
       1.57224583e+01, 1.46590174e+01, 1.72485553e+01, 1.23795969e+01,
       1.07718841e+01, 1.31195110e+01, 1.34701138e+01, 1.19729725e+01,
       1.14939859e+01, 8.61699893e+00, 1.81993745e+01, 8.03084207e+00,
       1.43389795e+01, 1.26234617e+01, 1.02323625e+01, 1.08058964e+01,
       8.30684210e+00, 1.69113916e+01, 1.00272636e+01, 9.70578557e+00,
       1.22888625e+01, 1.18936703e+01, 1.08153175e+01, 7.64653145e+00,
       8.42372731e+00, 1.27432597e+01, 8.81677827e+00, 1.12727308e+01,
       1.15057851e+01, 8.76618508e+00, 1.28855457e+01, 1.09988189e+01,
       9.00065834e+00, 1.15850565e+01, 1.22667487e+01, 1.32523855e+01,
       1.26562606e+01, 9.82894445e+00, 1.47517386e+01, 1.29841215e+01,
       1.04222145e+01, 1.47738410e+01, 9.48576097e+00, 9.70301755e+00,
       1.23668217e+01, 1.00376277e+01, 1.12186669e+01, 9.82033234e+00,
       7.25024408e+00, 9.34917664e+00, 1.03614516e+01, 8.43320635e+00,
       1.03952894e+01, 1.39845257e+01, 9.01739606e+00, 1.09542766e+01,
       6.23604935e+00, 9.50880834e+00, 1.14698870e+01, 1.27752970e+01,
       1.32912699e+01, 1.04783167e+01, 9.20214770e+00, 1.40564759e+01,
       1.28061153e+01, 1.12641397e+01, 1.14952156e+01, 9.60929971e+00,
       9.94190247e+00, 9.77008229e+00, 8.86634376e+00, 9.60557990e+00,
       1.50377205e+01, 1.14159841e+01, 1.11286807e+01, 9.85619383e+00,
       1.07369399e+01, 7.39393864e+00, 8.09483252e+00, 7.88759894e+00,
       1.22331807e+01, 1.33772211e+01, 8.65019937e+00, 1.03340761e+01,
       6.56523277e+00, 1.09166178e+01, 1.09103088e+01, 1.11619388e+01,
       1.16856102e+01, 8.58089582e+00, 1.14376590e+01, 9.16716903e+00,
       1.03144680e+01, 1.28723435e+01, 1.16472420e+01, 1.30442067e+01,
       1.25112125e+01, 1.44705674e+01, 9.64582021e+00, 8.85040727e+00,
       1.04339238e+01, 8.09723781e+00, 8.97048530e+00, 1.00334216e+01,
       1.68151897e+01, 8.50352193e+00, 1.10817559e+01, 7.70439284e+00,
       1.19263501e+01, 1.04877976e+01, 1.23025196e+01, 1.25773267e+01,
       1.01634365e+01, 1.36482363e+01, 1.00953851e+01, 7.79892696e+00,
       1.10483375e+01, 1.37800114e+01, 1.24220852e+01, 1.13631785e+01,
       1.33341061e+01, 1.18624560e+01, 7.50544262e+00, 1.11591374e+01,
       1.16519684e+01, 6.42915297e+00, 1.65935994e+01, 1.05347634e+01,
       1.11016189e+01, 7.20303622e+00, 1.32032593e+01, 1.27551715e+01,
       1.07628971e+01, 1.20507226e+01, 1.01515757e+01, 1.25629188e+01,
       1.27135188e+01, 1.23888396e+01, 1.23039621e+01, 9.76300912e+00,
       1.41578039e+01, 1.05108650e+01, 1.28047393e+01, 8.50522322e+00,
       1.36872538e+01, 1.02922394e+01, 8.17353486e+00, 1.02502447e+01,
       8.89865539e+00, 1.12922353e+01, 6.85692327e+00, 8.43205983e+00,
       1.20569779e+01, 9.94667109e+00, 9.25859907e+00, 9.39294633e+00,
       1.13368472e+01, 1.16309861e+01, 1.03370865e+01, 1.35128686e+01,
       1.15238053e+01, 1.15777386e+01, 1.15653087e+01, 1.33044823e+01,
       1.18282893e+01, 1.23876536e+01, 1.21344953e+01, 1.33095350e+01,
       1.34888776e+01, 1.25343126e+01, 9.20603956e+00, 1.31596955e+01,
       1.19617523e+01, 1.02531585e+01, 1.21278723e+01, 8.79384842e+00,
       1.02040591e+01, 1.18199512e+01, 1.05994860e+01, 1.11040801e+01,
       1.17993139e+01, 1.23456565e+01, 1.42804095e+01, 6.22729665e+00,
       1.48892196e+01, 1.12858838e+01, 1.07774335e+01, 1.27326671e+01,
       1.03467226e+01, 8.85134655e+00, 1.33718639e+01, 1.21783203e+01,
       8.80652431e+00, 1.43272882e+01, 1.04437424e+01, 1.24541156e+01,
       1.01356802e+01, 1.25731547e+01, 1.13427489e+01, 1.07334045e+01,
       1.02045639e+01, 6.38754054e+00, 1.23408242e+01, 1.05324355e+01,
       1.36591713e+01, 1.38241165e+01, 1.03844300e+01, 1.31476388e+01,
       9.44869967e+00, 6.59380803e+00, 1.06076313e+01, 1.00849980e+01,
       1.12272653e+01, 1.20155510e+01, 8.52984377e+00, 1.11507730e+01,
       1.29891782e+01, 1.03502356e+01, 1.21461955e+01, 1.34462426e+01,
       1.08974174e+01, 7.97002278e+00, 1.39813998e+01, 9.87670033e+00,
       8.01966602e+00, 8.69231600e+00, 1.17719052e+01, 1.58537810e+01,
       1.14693976e+01, 1.12007621e+01, 8.59494448e+00, 9.53394279e+00,
       1.32496690e+01, 1.28729447e+01, 1.16989187e+01, 1.04727762e+01,
       9.54522026e+00, 1.09018250e+01, 1.22705903e+01, 1.15376561e+01,
       1.08395495e+01, 1.15548815e+01, 1.40835903e+01, 7.18769235e+00,
       1.07986677e+01, 1.19638374e+01, 1.11914298e+01, 9.73679930e+00,
       1.37319059e+01, 8.24831267e+00, 1.18280372e+01, 9.51055489e+00,
       1.13544489e+01, 1.21494059e+01, 1.23387033e+01, 7.30325245e+00,
       1.03922033e+01, 1.12173444e+01, 1.17790507e+01, 1.22022540e+01,
       1.10428833e+01, 1.07003388e+01, 1.41234313e+01, 1.33727228e+01,
       1.12595474e+01, 9.67886367e+00, 1.24782485e+01, 8.64382095e+00,
       1.20900194e+01, 1.32257827e+01, 8.29433849e+00, 1.23609565e+01,
       1.14590980e+01, 9.32696405e+00, 7.27106916e+00, 1.16214143e+01,
       9.83095111e+00, 1.56727696e+01, 1.04713274e+01, 1.29125219e+01,
       1.11410488e+01, 9.84812657e+00, 8.47766630e+00, 1.28630526e+01,
       9.38568105e+00, 1.36319915e+01, 1.23981311e+01, 1.23521746e+01,
       1.10851702e+01, 1.21827383e+01, 8.67565301e+00, 1.10445528e+01,
       1.06877330e+01, 1.39168427e+01, 1.20459038e+01, 1.23463459e+01,
       1.00573793e+01, 1.07946230e+01, 1.05466488e+01, 9.82681953e+00,
       1.02558479e+01, 1.08196255e+01, 8.85507385e+00, 1.03311225e+01,
       1.07474917e+01, 1.04750232e+01, 9.75471592e+00, 1.07368923e+01,
       1.18850016e+01, 7.56249586e+00, 1.16761225e+01, 1.35389865e+01,
       1.14089117e+01, 1.32819223e+01, 1.08252157e+01, 1.24917300e+01,
       1.06142180e+01, 9.58440917e+00, 1.18943890e+01, 9.12690404e+00,
       8.31345333e+00, 1.21081771e+01, 1.38173415e+01, 1.31011532e+01,
       9.47850128e+00, 1.25777290e+01, 8.82952259e+00, 1.01515770e+01,
       1.10068933e+01, 1.23728202e+01, 1.19011866e+01, 9.23355184e+00,
       8.61784533e+00, 9.73945856e+00, 1.13767438e+01, 1.08851710e+01,
       1.67527688e+01, 1.18940518e+01, 1.23328816e+01, 1.13913004e+01,
       1.41431110e+01, 1.40992800e+01, 1.16550781e+01, 8.17277372e+00,
       9.58685496e+00, 1.29730909e+01, 1.10646816e+01, 1.03424477e+01,
       1.01589409e+01, 1.05285107e+01, 1.73525367e+01, 1.10486581e+01,
       1.22689357e+01, 9.63416839e+00, 1.08387007e+01, 1.50798355e+01,
       1.17630521e+01, 9.55314575e+00, 1.34816817e+01, 1.17262834e+01,
       1.19850462e+01, 1.35296775e+01, 9.97253987e+00, 8.53815024e+00,
       1.20504384e+01, 1.01488831e+01, 1.01568981e+01, 9.89152147e+00,
       1.17952535e+01, 7.67472300e+00, 8.76739683e+00, 1.30269825e+01,
       9.88764379e+00, 6.13901385e+00, 1.20075068e+01, 7.39560993e+00,
       1.32172416e+01, 1.20745950e+01, 1.00798393e+01, 1.31408646e+01,
       1.30618256e+01, 1.16644509e+01, 9.97726888e+00, 1.34545997e+01,
       1.48438095e+01, 1.05890835e+01, 1.24280030e+01, 1.10005700e+01,
       1.44892371e+01, 9.14173548e+00, 1.29909724e+01, 1.21958195e+01,
       1.09879532e+01, 1.15680388e+01, 5.77883922e+00, 1.30903348e+01,
       1.12351537e+01, 9.68438417e+00, 1.37731457e+01, 1.32506398e+01,
       1.18380981e+01, 1.11169445e+01, 1.28657217e+01, 1.31993464e+01,
       1.23082674e+01, 1.42628891e+01, 1.35206666e+01, 9.34093193e+00,
       7.57032547e+00, 1.28418835e+01, 1.13553405e+01, 1.00184639e+01,
       1.25992503e+01, 8.81296671e+00, 1.02268860e+01, 9.47108103e+00,
       8.06955265e+00, 9.20394602e+00, 1.35948136e+01, 1.32272874e+01,
       1.10822536e+01, 1.18316938e+01, 1.38198696e+01, 8.72868133e+00,
       9.93094333e+00, 8.00132333e+00, 9.27760125e+00, 1.04041378e+01,
       1.10120040e+01, 1.05786738e+01, 1.14981601e+01, 1.31821080e+01,
       1.02766971e+01, 1.38224350e+01, 8.43098578e+00, 8.58474359e+00,
       1.38477866e+01, 1.48391542e+01, 9.07216084e+00, 1.48539461e+01,
       1.05448841e+01, 1.46129072e+01, 8.52043533e+00, 9.79855950e+00,
       1.54438847e+01, 9.01027487e+00, 1.30794643e+01, 9.39956983e+00,
       1.28729728e+01, 1.24788227e+01, 1.14150091e+01, 7.52573442e+00,
       1.02338848e+01, 1.11049328e+01, 9.79856610e+00, 1.23887419e+01,
       1.30597411e+01, 1.11794661e+01, 9.72944274e+00])
```

In [ ]:
```
X_test
```

Out[ ]:
```
array([[0.665, 0.53 , 0.185, ..., 1.   , 0.   , 0.   ],
       [0.545, 0.41 , 0.12 , ..., 0.   , 0.   , 1.   ],
       [0.475, 0.37 , 0.125, ..., 0.   , 0.   , 1.   ],
       ...,
       [0.59 , 0.455, 0.175, ..., 1.   , 0.   , 0.   ],
       [0.585, 0.455, 0.155, ..., 0.   , 0.   , 1.   ],
       [0.42 , 0.325, 0.115, ..., 0.   , 1.   , 0.   ]])
```

In [ ]:
```
y_test
```

Out[ ]:
```
3204    16.5
1121    10.5
947     10.5
3184    14.5
3384     8.5
        ...
2571    10.5
2943    11.5
2137    11.5
4008    13.5
2734     9.5
Name: age, Length: 999, dtype: float64
```

In [ ]:
```
p = mean_squared_error(y_test, y_test_pred)
print('Mean Squared error of testing set :%2f'%p)
```

```
Mean Squared error of testing set :3.772386
```

14. Measure the performance using Metrics.

In [ ]:
```
from sklearn.metrics import r2_score
s = r2_score(y_train, y_train_pred)
print('R2 Score of training set:%.2f'%s)
```

```
R2 Score of training set:0.53
```

In [ ]:
```
from sklearn.metrics import r2_score
p = r2_score(y_test, y_test_pred)
print('R2 Score of testing set:%.2f'%p)
```

```
R2 Score of testing set:0.55
```