DESCRIPTION

Background of Problem Statement :

The US Census Bureau has published California Census Data which has 10 types of metrics such as the population, median income, median housing price, and so on for each block group in California. The dataset also serves as an input for project scoping and tries to specify the functional and nonfunctional requirements for it.

Problem Objective :

The project aims at building a model of housing prices to predict median house values in California using the provided dataset. This model should learn from the data and be able to predict the median housing price in any district, given all the other metrics.

Districts or block groups are the smallest geographical units for which the US Census Bureau publishes sample data (a block group typically has a population of 600 to 3,000 people). There are 20,640 districts in the project dataset.

[40]:
```
#Import Necessary Libraries:
import pandas as pd
import numpy as np

from sklearn.preprocessing import LabelEncoder,StandardScaler
from sklearn.linear_model import LinearRegression,Ridge,Lasso,ElasticNet
from sklearn.tree import DecisionTreeRegressor
import statsmodels.formula.api as smf

from sklearn.metrics import mean_squared_error,r2_score
from math import sqrt

import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

import warnings
warnings.filterwarnings('ignore')

from matplotlib.axes._axes import _log as matplotlib_axes_logger
matplotlib_axes_logger.setLevel('ERROR')
```

Build a model of housing prices to predict median house values in California using the provided dataset.

Load the data :

Read the "housing.csv" file from the folder into the program.

Print first few rows of this data.

[41]:
```
df_house=pd.read_excel("1553768847_housing.xlsx")
```

[42]:
```
df_house.head()
```

[42]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	ocean_proximity	median_house_value
0	-122.23	37.88	41	880	129.0	322	126	8.3252	NEAR BAY	452600
1	-122.22	37.86	21	7099	1106.0	2401	1138	8.3014	NEAR BAY	358500
2	-122.24	37.85	52	1467	190.0	496	177	7.2574	NEAR BAY	352100
3	-122.25	37.85	52	1274	235.0	558	219	5.6431	NEAR BAY	341300
4	-122.25	37.85	52	1627	280.0	565	259	3.8462	NEAR BAY	342200

[43]:
```
import math
print(math.log(452600))
```

```
13.022764012181574
```

[44]:
```
df_house.columns
```

[44]:
```
Index(['longitude', 'latitude', 'housing_median_age', 'total_rooms',
       'total_bedrooms', 'population', 'households', 'median_income',
       'ocean_proximity', 'median_house_value'],
      dtype='object')
```

Handle missing values :

Fill the missing values with the mean of the respective column.

[45]:
```
df_house.isnull().sum()
```

[45]:
```
longitude               0
latitude                0
housing_median_age      0
total_rooms             0
total_bedrooms        207
population              0
households              0
median_income           0
ocean_proximity         0
median_house_value      0
dtype: int64
```

[46]:
```
df_house.total_bedrooms=df_house.total_bedrooms.fillna(df_house.total_bedrooms.mean())
df_house.isnull().sum()
```

[46]:
```
longitude             0
latitude              0
housing_median_age    0
total_rooms           0
total_bedrooms        0
population            0
households            0
median_income         0
ocean_proximity       0
median_house_value    0
dtype: int64
```

Encode categorical data :

Convert categorical column in the dataset to numerical data.

Convert categorical column in the dataset to numerical data.

```
[47]: le = LabelEncoder()
      df_house['ocean_proximity']=le.fit_transform(df_house['ocean_proximity'])
```

Standardize data :

Standardize training and test datasets.

```
[52]: names = df_house.columns

      scaler = StandardScaler()

      scaled_df = scaler.fit_transform(df_house)
      scaled_df = pd.DataFrame(scaled_df, columns=names)
      scaled_df.head()
```

[52]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	ocean_proximity	median_house_value
0	-1.327835	1.052548	0.982143	-0.804819	-0.975228	-0.974429	-0.977033	2.344766	1.291089	2.129631
1	-1.322844	1.043185	-0.607019	2.045890	1.355088	0.861439	1.669961	2.332238	1.291089	1.314156
2	-1.332827	1.038503	1.856182	-0.535746	-0.829732	-0.820777	-0.843637	1.782699	1.291089	1.258693
3	-1.337818	1.038503	1.856182	-0.624215	-0.722399	-0.766028	-0.733781	0.932968	1.291089	1.165100
4	-1.337818	1.038503	1.856182	-0.462404	-0.615066	-0.759847	-0.629157	-0.012881	1.291089	1.172900

Extract input (X) and output (Y) data from the dataset.

```
[53]: X_Features=['longitude', 'latitude', 'housing_median_age', 'total_rooms',
            'total_bedrooms', 'population', 'households', 'median_income',
            'ocean_proximity']
      X=scaled_df[X_Features]
      Y=scaled_df['median_house_value']

      print(type(X))
      print(type(Y))
```

```
<class 'pandas.core.frame.DataFrame'>
<class 'pandas.core.series.Series'>
```

Split the dataset :

```
[54]: 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=1)

      print (x_train.shape, y_train.shape)
      print (x_test.shape, y_test.shape)
```

```
(16512, 9) (16512,)
(4128, 9) (4128,)
```

Perform Linear Regression :

Perform Linear Regression on training data.

Predict output for test dataset using the fitted model.

Print root mean squared error (RMSE) from Linear Regression. [HINT: Import mean_squared_error from sklearn.metrics]

```
[55]: linreg=LinearRegression()
      linreg.fit(x_train,y_train)
```

```
[55]: LinearRegression()
```

```
[56]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
[56]: LinearRegression(normalize=False)
```

```
[57]: y_predict = linreg.predict(x_test)
```

```
[58]: print(sqrt(mean_squared_error(y_test,y_predict)))
      print((r2_score(y_test,y_predict)))
```

```
0.6056598120301221
0.6276223517950295
```

Perform Decision Tree Regression :

Perform Decision Tree Regression on training data.

Predict output for test dataset using the fitted model.

Print root mean squared error from Decision Tree Regression.

```
[59]: dtreg=DecisionTreeRegressor()
      dtreg.fit(x_train,y_train)
```

```
[59]: DecisionTreeRegressor()
```

```
[61]: y_predict = dtreg.predict(x_test)
      print(sqrt(mean_squared_error(y_test,y_predict)))
      print((r2_score(y_test,y_predict)))
```

```
0.5967359681619542
0.6385147998268702
```

Perform Random Forest Regression :

Perform Random Forest Regression on training data.

Predict output for test dataset using the fitted model.

Print RMSE (root mean squared error) from Random Forest Regression.

```
[63]: from sklearn.ensemble import RandomForestRegressor
      rfreg=RandomForestRegressor()
```

```
[63]: RandomForestRegressor()
```

```
[66]: rfreg=RandomForestRegressor()
      rfreg.fit(x_train,y_train)
```

```
[66]: RandomForestRegressor()
```

```
[69]: y_predict = rfreg.predict(x_test)
      print(sqrt(mean_squared_error(y_test,y_predict)))
      print((r2_score(y_test,y_predict)))

0.4256352518189698
0.8160917256041964
```

Perform Linear Regression with one independent variable :

Extract just the median_income column from the independent variables (from X_train and X_test).

Perform Linear Regression to predict housing values based on median_income.

Predict output for test dataset using the fitted model.

Plot the fitted model for training data as well as for test data to check if the fitted model satisfies the test data.

```
[70]: x_train_Income=x_train[['median_income']]
      x_test_Income=x_test[['median_income']]
```

```
[71]: print(x_train_Income.shape)
      print(y_train.shape)

(16512, 1)
(16512,)
```

```
[72]: linreg=LinearRegression()
      linreg.fit(x_train_Income,y_train)
      y_predict = linreg.predict(x_test_Income)
```

```
[73]: print(linreg.intercept_, linreg.coef_)
      print(sqrt(mean_squared_error(y_test,y_predict)))
      print((r2_score(y_test,y_predict)))

0.005623019866893164 [0.69238221]
0.7212595914243148
0.47190835934467734
```

```
[74]: scaled_df.plot(kind='scatter',x='median_income',y='median_house_value')
      plt.plot(x_test_Income,y_predict,c='red',linewidth=2)
```

```
[74]: [<matplotlib.lines.Line2D at 0x7f46e10bb290>]
```