{

"cells": [

{

"cell\_type": "code",

"execution\_count": 1,

"metadata": {},

"outputs": [],

"source": [

"import pandas as pd\n",

"import numpy as np\n",

"import matplotlib.pyplot as plt\n",

"%matplotlib inline\n",

"import seaborn as sns"

]

},

{

"cell\_type": "code",

"execution\_count": 2,

"metadata": {

"collapsed": true,

"jupyter": {

"outputs\_hidden": true

}

},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"<class 'pandas.core.frame.DataFrame'>\n",

"RangeIndex: 20640 entries, 0 to 20639\n",

"Data columns (total 10 columns):\n",

" # Column Non-Null Count Dtype \n",

"--- ------ -------------- ----- \n",

" 0 longitude 20640 non-null float64\n",

" 1 latitude 20640 non-null float64\n",

" 2 housing\_median\_age 20640 non-null int64 \n",

" 3 total\_rooms 20640 non-null int64 \n",

" 4 total\_bedrooms 20433 non-null float64\n",

" 5 population 20640 non-null int64 \n",

" 6 households 20640 non-null int64 \n",

" 7 median\_income 20640 non-null float64\n",

" 8 ocean\_proximity 20640 non-null object \n",

" 9 median\_house\_value 20640 non-null int64 \n",

"dtypes: float64(4), int64(5), object(1)\n",

"memory usage: 1.6+ MB\n"

]

},

{

"data": {

"text/html": [

"<div>\n",

"<style scoped>\n",

" .dataframe tbody tr th:only-of-type {\n",

" vertical-align: middle;\n",

" }\n",

"\n",

" .dataframe tbody tr th {\n",

" vertical-align: top;\n",

" }\n",

"\n",

" .dataframe thead th {\n",

" text-align: right;\n",

" }\n",

"</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",

" <thead>\n",

" <tr style=\"text-align: right;\">\n",

" <th></th>\n",

" <th>longitude</th>\n",

" <th>latitude</th>\n",

" <th>housing\_median\_age</th>\n",

" <th>total\_rooms</th>\n",

" <th>total\_bedrooms</th>\n",

" <th>population</th>\n",

" <th>households</th>\n",

" <th>median\_income</th>\n",

" <th>ocean\_proximity</th>\n",

" <th>median\_house\_value</th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>0</th>\n",

" <td>-122.23</td>\n",

" <td>37.88</td>\n",

" <td>41</td>\n",

" <td>880</td>\n",

" <td>129.0</td>\n",

" <td>322</td>\n",

" <td>126</td>\n",

" <td>8.3252</td>\n",

" <td>NEAR BAY</td>\n",

" <td>452600</td>\n",

" </tr>\n",

" <tr>\n",

" <th>1</th>\n",

" <td>-122.22</td>\n",

" <td>37.86</td>\n",

" <td>21</td>\n",

" <td>7099</td>\n",

" <td>1106.0</td>\n",

" <td>2401</td>\n",

" <td>1138</td>\n",

" <td>8.3014</td>\n",

" <td>NEAR BAY</td>\n",

" <td>358500</td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"</div>"

],

"text/plain": [

" longitude latitude housing\_median\_age total\_rooms total\_bedrooms \\\n",

"0 -122.23 37.88 41 880 129.0 \n",

"1 -122.22 37.86 21 7099 1106.0 \n",

"\n",

" population households median\_income ocean\_proximity median\_house\_value \n",

"0 322 126 8.3252 NEAR BAY 452600 \n",

"1 2401 1138 8.3014 NEAR BAY 358500 "

]

},

"execution\_count": 2,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"df=pd.read\_excel(\"1553768847\_housing.xlsx\")\n",

"df.head()\n",

"df.shape\n",

"df.info()\n",

"df.describe()\n",

"df.head(2)"

]

},

{

"cell\_type": "code",

"execution\_count": 7,

"metadata": {},

"outputs": [

{

"data": {

"text/html": [

"<div>\n",

"<style scoped>\n",

" .dataframe tbody tr th:only-of-type {\n",

" vertical-align: middle;\n",

" }\n",

"\n",

" .dataframe tbody tr th {\n",

" vertical-align: top;\n",

" }\n",

"\n",

" .dataframe thead th {\n",

" text-align: right;\n",

" }\n",

"</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",

" <thead>\n",

" <tr style=\"text-align: right;\">\n",

" <th></th>\n",

" <th>longitude</th>\n",

" <th>latitude</th>\n",

" <th>housing\_median\_age</th>\n",

" <th>total\_rooms</th>\n",

" <th>total\_bedrooms</th>\n",

" <th>population</th>\n",

" <th>households</th>\n",

" <th>median\_income</th>\n",

" <th>median\_house\_value</th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>count</th>\n",

" <td>20640.000000</td>\n",

" <td>20640.000000</td>\n",

" <td>20640.000000</td>\n",

" <td>20640.000000</td>\n",

" <td>20640.000000</td>\n",

" <td>20640.000000</td>\n",

" <td>20640.000000</td>\n",

" <td>20640.000000</td>\n",

" <td>20640.000000</td>\n",

" </tr>\n",

" <tr>\n",

" <th>mean</th>\n",

" <td>-119.569704</td>\n",

" <td>35.631861</td>\n",

" <td>28.639486</td>\n",

" <td>2635.763081</td>\n",

" <td>537.870553</td>\n",

" <td>1425.476744</td>\n",

" <td>499.539680</td>\n",

" <td>3.870671</td>\n",

" <td>206855.816909</td>\n",

" </tr>\n",

" <tr>\n",

" <th>std</th>\n",

" <td>2.003532</td>\n",

" <td>2.135952</td>\n",

" <td>12.585558</td>\n",

" <td>2181.615252</td>\n",

" <td>419.266592</td>\n",

" <td>1132.462122</td>\n",

" <td>382.329753</td>\n",

" <td>1.899822</td>\n",

" <td>115395.615874</td>\n",

" </tr>\n",

" <tr>\n",

" <th>min</th>\n",

" <td>-124.350000</td>\n",

" <td>32.540000</td>\n",

" <td>1.000000</td>\n",

" <td>2.000000</td>\n",

" <td>1.000000</td>\n",

" <td>3.000000</td>\n",

" <td>1.000000</td>\n",

" <td>0.499900</td>\n",

" <td>14999.000000</td>\n",

" </tr>\n",

" <tr>\n",

" <th>25%</th>\n",

" <td>-121.800000</td>\n",

" <td>33.930000</td>\n",

" <td>18.000000</td>\n",

" <td>1447.750000</td>\n",

" <td>297.000000</td>\n",

" <td>787.000000</td>\n",

" <td>280.000000</td>\n",

" <td>2.563400</td>\n",

" <td>119600.000000</td>\n",

" </tr>\n",

" <tr>\n",

" <th>50%</th>\n",

" <td>-118.490000</td>\n",

" <td>34.260000</td>\n",

" <td>29.000000</td>\n",

" <td>2127.000000</td>\n",

" <td>438.000000</td>\n",

" <td>1166.000000</td>\n",

" <td>409.000000</td>\n",

" <td>3.534800</td>\n",

" <td>179700.000000</td>\n",

" </tr>\n",

" <tr>\n",

" <th>75%</th>\n",

" <td>-118.010000</td>\n",

" <td>37.710000</td>\n",

" <td>37.000000</td>\n",

" <td>3148.000000</td>\n",

" <td>643.250000</td>\n",

" <td>1725.000000</td>\n",

" <td>605.000000</td>\n",

" <td>4.743250</td>\n",

" <td>264725.000000</td>\n",

" </tr>\n",

" <tr>\n",

" <th>max</th>\n",

" <td>-114.310000</td>\n",

" <td>41.950000</td>\n",

" <td>52.000000</td>\n",

" <td>39320.000000</td>\n",

" <td>6445.000000</td>\n",

" <td>35682.000000</td>\n",

" <td>6082.000000</td>\n",

" <td>15.000100</td>\n",

" <td>500001.000000</td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"</div>"

],

"text/plain": [

" longitude latitude housing\_median\_age total\_rooms \\\n",

"count 20640.000000 20640.000000 20640.000000 20640.000000 \n",

"mean -119.569704 35.631861 28.639486 2635.763081 \n",

"std 2.003532 2.135952 12.585558 2181.615252 \n",

"min -124.350000 32.540000 1.000000 2.000000 \n",

"25% -121.800000 33.930000 18.000000 1447.750000 \n",

"50% -118.490000 34.260000 29.000000 2127.000000 \n",

"75% -118.010000 37.710000 37.000000 3148.000000 \n",

"max -114.310000 41.950000 52.000000 39320.000000 \n",

"\n",

" total\_bedrooms population households median\_income \\\n",

"count 20640.000000 20640.000000 20640.000000 20640.000000 \n",

"mean 537.870553 1425.476744 499.539680 3.870671 \n",

"std 419.266592 1132.462122 382.329753 1.899822 \n",

"min 1.000000 3.000000 1.000000 0.499900 \n",

"25% 297.000000 787.000000 280.000000 2.563400 \n",

"50% 438.000000 1166.000000 409.000000 3.534800 \n",

"75% 643.250000 1725.000000 605.000000 4.743250 \n",

"max 6445.000000 35682.000000 6082.000000 15.000100 \n",

"\n",

" median\_house\_value \n",

"count 20640.000000 \n",

"mean 206855.816909 \n",

"std 115395.615874 \n",

"min 14999.000000 \n",

"25% 119600.000000 \n",

"50% 179700.000000 \n",

"75% 264725.000000 \n",

"max 500001.000000 "

]

},

"execution\_count": 7,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"df.describe()"

]

},

{

"cell\_type": "code",

"execution\_count": 20,

"metadata": {},

"outputs": [

{

"data": {

"text/plain": [

"13139 39320\n",

"10309 37937\n",

"9880 32627\n",

"6057 32054\n",

"12201 30450\n",

"9019 30405\n",

"12215 30401\n",

"922 28258\n",

"12623 27870\n",

"6066 27700\n",

"Name: total\_rooms, dtype: int64"

]

},

"execution\_count": 20,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"data.total\_rooms.sort\_values(ascending=False)[:10]"

]

},

{

"cell\_type": "code",

"execution\_count": 24,

"metadata": {},

"outputs": [

{

"data": {

"text/plain": [

"longitude 0\n",

"latitude 0\n",

"housing\_median\_age 0\n",

"total\_rooms 0\n",

"total\_bedrooms 0\n",

"population 0\n",

"households 0\n",

"median\_income 0\n",

"ocean\_proximity 0\n",

"median\_house\_value 0\n",

"dtype: int64"

]

},

"execution\_count": 24,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"df.isnull().sum()\n",

"df.total\_bedrooms\n",

"mode\_bedroom=df.total\_bedrooms.mode()\n",

"df.total\_bedrooms.fillna(value=mean\_bedroom,inplace=True)\n",

"df.isnull().sum()"

]

},

{

"cell\_type": "code",

"execution\_count": 25,

"metadata": {},

"outputs": [

{

"data": {

"image/png": "\n",

"text/plain": [

"<Figure size 1152x720 with 2 Axes>"

]

},

"metadata": {

"needs\_background": "light"

},

"output\_type": "display\_data"

}

],

"source": [

"corrmat=df.corr()\n",

"corrmat\n",

"m=np.ones\_like(corrmat)\n",

"m[np.tril\_indices\_from(m)]=0\n",

"plt.figure(figsize=(16,10))\n",

"sns.heatmap(corrmat,annot=True, mask=m)\n",

"plt.savefig(\"Corrmat.png\")"

]

},

{

"cell\_type": "code",

"execution\_count": 5,

"metadata": {},

"outputs": [

{

"data": {

"text/plain": [

"<1H OCEAN 9136\n",

"INLAND 6551\n",

"NEAR OCEAN 2658\n",

"NEAR BAY 2290\n",

"ISLAND 5\n",

"Name: ocean\_proximity, dtype: int64"

]

},

"execution\_count": 5,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"df.ocean\_proximity.value\_counts()"

]

},

{

"cell\_type": "code",

"execution\_count": 26,

"metadata": {},

"outputs": [],

"source": [

"df.ocean\_proximity.value\_counts()\n",

"dummies=pd.get\_dummies(df.ocean\_proximity)\n",

"dummies.head()\n",

"data=pd.concat([df,dummies.drop(columns=\"ISLAND\")],axis=\"columns\")\n",

"data.head()\n",

"data1=data.drop(columns=\"ocean\_proximity\")"

]

},

{

"cell\_type": "code",

"execution\_count": 22,

"metadata": {},

"outputs": [

{

"data": {

"text/html": [

"<div>\n",

"<style scoped>\n",

" .dataframe tbody tr th:only-of-type {\n",

" vertical-align: middle;\n",

" }\n",

"\n",

" .dataframe tbody tr th {\n",

" vertical-align: top;\n",

" }\n",

"\n",

" .dataframe thead th {\n",

" text-align: right;\n",

" }\n",

"</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",

" <thead>\n",

" <tr style=\"text-align: right;\">\n",

" <th></th>\n",

" <th>longitude</th>\n",

" <th>latitude</th>\n",

" <th>housing\_median\_age</th>\n",

" <th>total\_rooms</th>\n",

" <th>total\_bedrooms</th>\n",

" <th>population</th>\n",

" <th>households</th>\n",

" <th>median\_income</th>\n",

" <th>median\_house\_value</th>\n",

" <th>&lt;1H OCEAN</th>\n",

" <th>INLAND</th>\n",

" <th>NEAR BAY</th>\n",

" <th>NEAR OCEAN</th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>0</th>\n",

" <td>-122.23</td>\n",

" <td>37.88</td>\n",

" <td>41</td>\n",

" <td>880</td>\n",

" <td>129.0</td>\n",

" <td>322</td>\n",

" <td>126</td>\n",

" <td>8.3252</td>\n",

" <td>452600</td>\n",

" <td>0</td>\n",

" <td>0</td>\n",

" <td>1</td>\n",

" <td>0</td>\n",

" </tr>\n",

" <tr>\n",

" <th>1</th>\n",

" <td>-122.22</td>\n",

" <td>37.86</td>\n",

" <td>21</td>\n",

" <td>7099</td>\n",

" <td>1106.0</td>\n",

" <td>2401</td>\n",

" <td>1138</td>\n",

" <td>8.3014</td>\n",

" <td>358500</td>\n",

" <td>0</td>\n",

" <td>0</td>\n",

" <td>1</td>\n",

" <td>0</td>\n",

" </tr>\n",

" <tr>\n",

" <th>2</th>\n",

" <td>-122.24</td>\n",

" <td>37.85</td>\n",

" <td>52</td>\n",

" <td>1467</td>\n",

" <td>190.0</td>\n",

" <td>496</td>\n",

" <td>177</td>\n",

" <td>7.2574</td>\n",

" <td>352100</td>\n",

" <td>0</td>\n",

" <td>0</td>\n",

" <td>1</td>\n",

" <td>0</td>\n",

" </tr>\n",

" <tr>\n",

" <th>3</th>\n",

" <td>-122.25</td>\n",

" <td>37.85</td>\n",

" <td>52</td>\n",

" <td>1274</td>\n",

" <td>235.0</td>\n",

" <td>558</td>\n",

" <td>219</td>\n",

" <td>5.6431</td>\n",

" <td>341300</td>\n",

" <td>0</td>\n",

" <td>0</td>\n",

" <td>1</td>\n",

" <td>0</td>\n",

" </tr>\n",

" <tr>\n",

" <th>4</th>\n",

" <td>-122.25</td>\n",

" <td>37.85</td>\n",

" <td>52</td>\n",

" <td>1627</td>\n",

" <td>280.0</td>\n",

" <td>565</td>\n",

" <td>259</td>\n",

" <td>3.8462</td>\n",

" <td>342200</td>\n",

" <td>0</td>\n",

" <td>0</td>\n",

" <td>1</td>\n",

" <td>0</td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"</div>"

],

"text/plain": [

" longitude latitude housing\_median\_age total\_rooms total\_bedrooms \\\n",

"0 -122.23 37.88 41 880 129.0 \n",

"1 -122.22 37.86 21 7099 1106.0 \n",

"2 -122.24 37.85 52 1467 190.0 \n",

"3 -122.25 37.85 52 1274 235.0 \n",

"4 -122.25 37.85 52 1627 280.0 \n",

"\n",

" population households median\_income median\_house\_value <1H OCEAN \\\n",

"0 322 126 8.3252 452600 0 \n",

"1 2401 1138 8.3014 358500 0 \n",

"2 496 177 7.2574 352100 0 \n",

"3 558 219 5.6431 341300 0 \n",

"4 565 259 3.8462 342200 0 \n",

"\n",

" INLAND NEAR BAY NEAR OCEAN \n",

"0 0 1 0 \n",

"1 0 1 0 \n",

"2 0 1 0 \n",

"3 0 1 0 \n",

"4 0 1 0 "

]

},

"execution\_count": 22,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"data1.head()"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"data1.head()\n",

"df.columns.nunique()"

]

},

{

"cell\_type": "code",

"execution\_count": 29,

"metadata": {},

"outputs": [

{

"data": {

"text/plain": [

"(20640, 12)"

]

},

"execution\_count": 29,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"data1.head()\n",

"df.columns.nunique()\n",

"X=data1.drop(columns=\"median\_house\_value\")\n",

"y=data1.median\_house\_value\n",

"X.shape"

]

},

{

"cell\_type": "code",

"execution\_count": 30,

"metadata": {},

"outputs": [],

"source": [

"from sklearn.model\_selection import train\_test\_split as split\n",

"X\_train,X\_test,y\_train,y\_test=split(X,y,test\_size=0.20,random\_state=123)"

]

},

{

"cell\_type": "code",

"execution\_count": 31,

"metadata": {},

"outputs": [

{

"data": {

"text/plain": [

"LinearRegression(copy\_X=True, fit\_intercept=True, n\_jobs=None, normalize=False)"

]

},

"execution\_count": 31,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"from sklearn.linear\_model import LinearRegression\n",

"lr=LinearRegression()\n",

"lr.fit(X\_train,y\_train)"

]

},

{

"cell\_type": "code",

"execution\_count": 32,

"metadata": {},

"outputs": [],

"source": [

"y\_pred=lr.predict(X\_test)"

]

},

{

"cell\_type": "code",

"execution\_count": 33,

"metadata": {},

"outputs": [

{

"data": {

"text/plain": [

"0.644755396216805"

]

},

"execution\_count": 33,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"from sklearn.metrics import r2\_score,mean\_squared\_error\n",

"score=r2\_score(y\_test,y\_pred)\n",

"score\n"

]

},

{

"cell\_type": "code",

"execution\_count": 36,

"metadata": {},

"outputs": [

{

"data": {

"text/plain": [

"68732.03768156999"

]

},

"execution\_count": 36,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"mse=mean\_squared\_error(y\_test,y\_pred)\n",

"mse\n",

"rmse=np.sqrt(mse)\n",

"rmse"

]

},

{

"cell\_type": "code",

"execution\_count": 37,

"metadata": {},

"outputs": [

{

"data": {

"text/html": [

"<table class=\"simpletable\">\n",

"<caption>OLS Regression Results</caption>\n",

"<tr>\n",

" <th>Dep. Variable:</th> <td>median\_house\_value</td> <th> R-squared: </th> <td> 0.645</td> \n",

"</tr>\n",

"<tr>\n",

" <th>Model:</th> <td>OLS</td> <th> Adj. R-squared: </th> <td> 0.645</td> \n",

"</tr>\n",

"<tr>\n",

" <th>Method:</th> <td>Least Squares</td> <th> F-statistic: </th> <td> 2502.</td> \n",

"</tr>\n",

"<tr>\n",

" <th>Date:</th> <td>Sun, 05 Jul 2020</td> <th> Prob (F-statistic):</th> <td> 0.00</td> \n",

"</tr>\n",

"<tr>\n",

" <th>Time:</th> <td>15:56:58</td> <th> Log-Likelihood: </th> <td>-2.0734e+05</td>\n",

"</tr>\n",

"<tr>\n",

" <th>No. Observations:</th> <td> 16512</td> <th> AIC: </th> <td>4.147e+05</td> \n",

"</tr>\n",

"<tr>\n",

" <th>Df Residuals:</th> <td> 16499</td> <th> BIC: </th> <td>4.148e+05</td> \n",

"</tr>\n",

"<tr>\n",

" <th>Df Model:</th> <td> 12</td> <th> </th> <td> </td> \n",

"</tr>\n",

"<tr>\n",

" <th>Covariance Type:</th> <td>nonrobust</td> <th> </th> <td> </td> \n",

"</tr>\n",

"</table>\n",

"<table class=\"simpletable\">\n",

"<tr>\n",

" <td></td> <th>coef</th> <th>std err</th> <th>t</th> <th>P>|t|</th> <th>[0.025</th> <th>0.975]</th> \n",

"</tr>\n",

"<tr>\n",

" <th>Intercept</th> <td>-2.227e+06</td> <td> 9.79e+04</td> <td> -22.750</td> <td> 0.000</td> <td>-2.42e+06</td> <td>-2.04e+06</td>\n",

"</tr>\n",

"<tr>\n",

" <th>ocean\_proximity[T.INLAND]</th> <td>-4.038e+04</td> <td> 1940.686</td> <td> -20.809</td> <td> 0.000</td> <td>-4.42e+04</td> <td>-3.66e+04</td>\n",

"</tr>\n",

"<tr>\n",

" <th>ocean\_proximity[T.ISLAND]</th> <td> 1.543e+05</td> <td> 3.08e+04</td> <td> 5.012</td> <td> 0.000</td> <td> 9.4e+04</td> <td> 2.15e+05</td>\n",

"</tr>\n",

"<tr>\n",

" <th>ocean\_proximity[T.NEAR BAY]</th> <td>-6573.3187</td> <td> 2137.861</td> <td> -3.075</td> <td> 0.002</td> <td>-1.08e+04</td> <td>-2382.881</td>\n",

"</tr>\n",

"<tr>\n",

" <th>ocean\_proximity[T.NEAR OCEAN]</th> <td> 4072.1048</td> <td> 1749.739</td> <td> 2.327</td> <td> 0.020</td> <td> 642.428</td> <td> 7501.782</td>\n",

"</tr>\n",

"<tr>\n",

" <th>longitude</th> <td>-2.643e+04</td> <td> 1134.447</td> <td> -23.300</td> <td> 0.000</td> <td>-2.87e+04</td> <td>-2.42e+04</td>\n",

"</tr>\n",

"<tr>\n",

" <th>latitude</th> <td>-2.525e+04</td> <td> 1118.140</td> <td> -22.583</td> <td> 0.000</td> <td>-2.74e+04</td> <td>-2.31e+04</td>\n",

"</tr>\n",

"<tr>\n",

" <th>housing\_median\_age</th> <td> 1024.7734</td> <td> 48.943</td> <td> 20.938</td> <td> 0.000</td> <td> 928.840</td> <td> 1120.707</td>\n",

"</tr>\n",

"<tr>\n",

" <th>total\_rooms</th> <td> -4.1071</td> <td> 0.853</td> <td> -4.816</td> <td> 0.000</td> <td> -5.779</td> <td> -2.435</td>\n",

"</tr>\n",

"<tr>\n",

" <th>total\_bedrooms</th> <td> 74.0036</td> <td> 6.897</td> <td> 10.729</td> <td> 0.000</td> <td> 60.484</td> <td> 87.523</td>\n",

"</tr>\n",

"<tr>\n",

" <th>population</th> <td> -41.6259</td> <td> 1.240</td> <td> -33.583</td> <td> 0.000</td> <td> -44.056</td> <td> -39.196</td>\n",

"</tr>\n",

"<tr>\n",

" <th>households</th> <td> 75.3960</td> <td> 7.786</td> <td> 9.684</td> <td> 0.000</td> <td> 60.135</td> <td> 90.657</td>\n",

"</tr>\n",

"<tr>\n",

" <th>median\_income</th> <td> 3.869e+04</td> <td> 372.561</td> <td> 103.852</td> <td> 0.000</td> <td> 3.8e+04</td> <td> 3.94e+04</td>\n",

"</tr>\n",

"</table>\n",

"<table class=\"simpletable\">\n",

"<tr>\n",

" <th>Omnibus:</th> <td>3712.359</td> <th> Durbin-Watson: </th> <td> 1.995</td> \n",

"</tr>\n",

"<tr>\n",

" <th>Prob(Omnibus):</th> <td> 0.000</td> <th> Jarque-Bera (JB): </th> <td>11935.012</td>\n",

"</tr>\n",

"<tr>\n",

" <th>Skew:</th> <td> 1.140</td> <th> Prob(JB): </th> <td> 0.00</td> \n",

"</tr>\n",

"<tr>\n",

" <th>Kurtosis:</th> <td> 6.485</td> <th> Cond. No. </th> <td>7.27e+05</td> \n",

"</tr>\n",

"</table><br/><br/>Warnings:<br/>[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.<br/>[2] The condition number is large, 7.27e+05. This might indicate that there are<br/>strong multicollinearity or other numerical problems."

],

"text/plain": [

"<class 'statsmodels.iolib.summary.Summary'>\n",

"\"\"\"\n",

" OLS Regression Results \n",

"==============================================================================\n",

"Dep. Variable: median\_house\_value R-squared: 0.645\n",

"Model: OLS Adj. R-squared: 0.645\n",

"Method: Least Squares F-statistic: 2502.\n",

"Date: Sun, 05 Jul 2020 Prob (F-statistic): 0.00\n",

"Time: 15:56:58 Log-Likelihood: -2.0734e+05\n",

"No. Observations: 16512 AIC: 4.147e+05\n",

"Df Residuals: 16499 BIC: 4.148e+05\n",

"Df Model: 12 \n",

"Covariance Type: nonrobust \n",

"=================================================================================================\n",

" coef std err t P>|t| [0.025 0.975]\n",

"-------------------------------------------------------------------------------------------------\n",

"Intercept -2.227e+06 9.79e+04 -22.750 0.000 -2.42e+06 -2.04e+06\n",

"ocean\_proximity[T.INLAND] -4.038e+04 1940.686 -20.809 0.000 -4.42e+04 -3.66e+04\n",

"ocean\_proximity[T.ISLAND] 1.543e+05 3.08e+04 5.012 0.000 9.4e+04 2.15e+05\n",

"ocean\_proximity[T.NEAR BAY] -6573.3187 2137.861 -3.075 0.002 -1.08e+04 -2382.881\n",

"ocean\_proximity[T.NEAR OCEAN] 4072.1048 1749.739 2.327 0.020 642.428 7501.782\n",

"longitude -2.643e+04 1134.447 -23.300 0.000 -2.87e+04 -2.42e+04\n",

"latitude -2.525e+04 1118.140 -22.583 0.000 -2.74e+04 -2.31e+04\n",

"housing\_median\_age 1024.7734 48.943 20.938 0.000 928.840 1120.707\n",

"total\_rooms -4.1071 0.853 -4.816 0.000 -5.779 -2.435\n",

"total\_bedrooms 74.0036 6.897 10.729 0.000 60.484 87.523\n",

"population -41.6259 1.240 -33.583 0.000 -44.056 -39.196\n",

"households 75.3960 7.786 9.684 0.000 60.135 90.657\n",

"median\_income 3.869e+04 372.561 103.852 0.000 3.8e+04 3.94e+04\n",

"==============================================================================\n",

"Omnibus: 3712.359 Durbin-Watson: 1.995\n",

"Prob(Omnibus): 0.000 Jarque-Bera (JB): 11935.012\n",

"Skew: 1.140 Prob(JB): 0.00\n",

"Kurtosis: 6.485 Cond. No. 7.27e+05\n",

"==============================================================================\n",

"\n",

"Warnings:\n",

"[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.\n",

"[2] The condition number is large, 7.27e+05. This might indicate that there are\n",

"strong multicollinearity or other numerical problems.\n",

"\"\"\""

]

},

"execution\_count": 37,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"train,test=split(df,test\_size=0.20,random\_state=123)\n",

"from statsmodels.formula.api import ols\n",

"l=df.drop(columns=[\"median\_house\_value\"]).columns\n",

"features=\"+\".join(l)\n",

"formula=\"median\_house\_value~\"+ features\n",

"ols\_=ols(formula,train).fit()\n",

"ols\_.summary()"

]

},

{

"cell\_type": "code",

"execution\_count": 38,

"metadata": {},

"outputs": [

{

"data": {

"text/plain": [

"array([0.6447554 , 0.63864606, 0.64365928, 0.64559444, 0.63533901])"

]

},

"execution\_count": 38,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"## Another Method\n",

"\n",

"from sklearn.model\_selection import ShuffleSplit\n",

"from sklearn.model\_selection import cross\_val\_score\n",

"cv=ShuffleSplit(n\_splits=5,test\_size=.20,random\_state=123)\n",

"score=cross\_val\_score(LinearRegression(),X,y,cv=cv)\n",

"score"

]

},

{

"cell\_type": "code",

"execution\_count": 39,

"metadata": {},

"outputs": [

{

"data": {

"text/plain": [

"68725.5360219543"

]

},

"execution\_count": 39,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"##Using Ridge Regression\n",

"from sklearn.linear\_model import Ridge\n",

"\n",

"df\_x=data1.drop(columns=\"median\_house\_value\")\n",

"X=df\_x.values\n",

"y=data1.median\_house\_value.values\n",

"lridge=Ridge(alpha=2)\n",

"lridge.fit(X,y)\n",

"y\_pred=lridge.predict(X)\n",

"score=np.sqrt(mean\_squared\_error(y,y\_pred))\n",

"score"

]

},

{

"cell\_type": "code",

"execution\_count": 40,

"metadata": {},

"outputs": [

{

"data": {

"text/plain": [

"0.6452857018458292"

]

},

"execution\_count": 40,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"lridge.score(X,y)"

]

},

{

"cell\_type": "code",

"execution\_count": 41,

"metadata": {},

"outputs": [

{

"name": "stderr",

"output\_type": "stream",

"text": [

"/usr/local/lib/python3.7/site-packages/sklearn/linear\_model/\_coordinate\_descent.py:476: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations. Duality gap: 46249659376009.54, tolerance: 27483198193.688515\n",

" positive)\n"

]

},

{

"data": {

"text/plain": [

"0.6452990302612521"

]

},

"execution\_count": 41,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"## Using Lasso Regression\n",

"\n",

"from sklearn.linear\_model import Lasso\n",

"\n",

"df\_x=data1.drop(columns=\"median\_house\_value\")\n",

"X=df\_x.values\n",

"y=data1.median\_house\_value.values\n",

"lasso=Lasso(alpha=2.0)\n",

"lasso.fit(X,y)\n",

"y\_pred=lasso.predict(X)\n",

"score=np.sqrt(mean\_squared\_error(y,y\_pred))\n",

"score\n",

"lasso.score(X,y)"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"## Using Decision Tree Regressor\n",

"\n",

"from sklearn.tree import DecisionTreeRegressor\n",

"\n",

"df\_x=data1.drop(columns=\"median\_house\_value\")\n",

"X=df\_x.values\n",

"y=data1.median\_house\_value.values\n",

"dtr=DecisionTreeRegressor()\n",

"dtr.fit(X,y)\n",

"y\_pred=dtr.predict(X)\n",

"score=np.sqrt(mean\_squared\_error(y,y\_pred))\n",

"score\n",

"dtr.score(X,y)\n"

]

},

{

"cell\_type": "code",

"execution\_count": 44,

"metadata": {},

"outputs": [],

"source": [

"## Linear Regression with One independent variable\n",

"\n",

"X\_train\_=pd.DataFrame(X\_train.median\_income)\n",

"X\_test\_=pd.DataFrame(X\_test.median\_income)"

]

},

{

"cell\_type": "code",

"execution\_count": 45,

"metadata": {},

"outputs": [

{

"ename": "NameError",

"evalue": "name 'y\_predict' is not defined",

"output\_type": "error",

"traceback": [

"\u001b[0;31m---------------------------------------------------------------------------\u001b[0m",

"\u001b[0;31mNameError\u001b[0m Traceback (most recent call last)",

"\u001b[0;32m<ipython-input-45-fb4b1583a186>\u001b[0m in \u001b[0;36m<module>\u001b[0;34m\u001b[0m\n\u001b[1;32m 4\u001b[0m \u001b[0mlr\u001b[0m\u001b[0;34m.\u001b[0m\u001b[0mscore\u001b[0m\u001b[0;34m(\u001b[0m\u001b[0mX\_test\_\u001b[0m\u001b[0;34m,\u001b[0m\u001b[0my\_test\u001b[0m\u001b[0;34m)\u001b[0m\u001b[0;34m\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n\u001b[1;32m 5\u001b[0m \u001b[0mr2\u001b[0m\u001b[0;34m=\u001b[0m\u001b[0mr2\_score\u001b[0m\u001b[0;34m(\u001b[0m\u001b[0my\_test\u001b[0m\u001b[0;34m,\u001b[0m\u001b[0my\_predict\_\u001b[0m\u001b[0;34m)\u001b[0m\u001b[0;34m\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n\u001b[0;32m----> 6\u001b[0;31m \u001b[0mRMSE\u001b[0m\u001b[0;34m=\u001b[0m\u001b[0mnp\u001b[0m\u001b[0;34m.\u001b[0m\u001b[0msqrt\u001b[0m\u001b[0;34m(\u001b[0m\u001b[0mmean\_squared\_error\u001b[0m\u001b[0;34m(\u001b[0m\u001b[0my\_test\u001b[0m\u001b[0;34m,\u001b[0m\u001b[0my\_predict\u001b[0m\u001b[0;34m)\u001b[0m\u001b[0;34m)\u001b[0m\u001b[0;34m\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n\u001b[0m",

"\u001b[0;31mNameError\u001b[0m: name 'y\_predict' is not defined"

]

}

],

"source": [

"lr=LinearRegression()\n",

"model=lr.fit(X\_train\_,y\_train)\n",

"y\_predict\_=lr.predict(X\_test\_)\n",

"lr.score(X\_test\_,y\_test)\n",

"r2=r2\_score(y\_test,y\_predict\_)\n",

"RMSE=np.sqrt(mean\_squared\_error(y\_test,y\_predict))"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": []

}

],

"metadata": {

"kernelspec": {

"display\_name": "Python 3",

"language": "python",

"name": "python3"

},

"language\_info": {

"codemirror\_mode": {

"name": "ipython",

"version": 3

},

"file\_extension": ".py",

"mimetype": "text/x-python",

"name": "python",

"nbconvert\_exporter": "python",

"pygments\_lexer": "ipython3",

"version": "3.7.6"

}

},

"nbformat": 4,

"nbformat\_minor": 4

}