California Housing Price Prediction

Xiang Liu   94457
Lan Zhang 94807

California Housing Price Prediction

Topic

To predict the median housing prices in different administrative regions of California.

Data Resources

California House Price Database

(Kaggle)

Link:
http://raw.githubusercontent.com/ageron/handson-ml/master/datasets/housing/housing.tgz

Tools and Method

Python

LinearRegression

DecisionTreeRegression

Raw Data Sample

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_proximity
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	452600.0	NEAR BAY
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	358500.0	NEAR BAY
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	352100.0	NEAR BAY
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	341300.0	NEAR BAY
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	342200.0	NEAR BAY
5	-122.25	37.85	52.0	919.0	213.0	413.0	193.0	4.0368	269700.0	NEAR BAY

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value
count	20640.000000	20640.000000	20640.000000	20640.000000	20433.000000	20640.000000	20640.000000	20640.000000	20640.000000
mean	-119.569704	35.631861	28.639486	2635.763081	537.870553	1425.476744	499.539680	3.870671	206855.816909
std	2.003532	2.135952	12.585558	2181.615252	421.385070	1132.462122	382.329753	1.899822	115395.615874
min	-124.350000	32.540000	1.000000	2.000000	1.000000	3.000000	1.000000	0.499900	14999.000000
25%	-121.800000	33.930000	18.000000	1447.750000	296.000000	787.000000	280.000000	2.563400	119600.000000
50%	-118.490000	34.260000	29.000000	2127.000000	435.000000	1166.000000	409.000000	3.534800	179700.000000
75%	-118.010000	37.710000	37.000000	3148.000000	647.000000	1725.000000	605.000000	4.743250	264725.000000
max	-114.310000	41.950000	52.000000	39320.000000	6445.000000	35682.000000	6082.000000	15.000100	500001.000000

Data Types

```
data.dtypes
```

```
longitude             float64
latitude              float64
housing_median_age    float64
total_rooms           float64
total_bedrooms        float64
population            float64
households            float64
median_income         float64
median_house_value    float64
ocean_proximity        object
dtype: object
```

Process

Data Cleaning

Working with data

Clean the data

(missing data, outliers)

80% train, 20% test

Standardize the data

Generate a model

Machine Learning Algorithm

Explore the traits of the data and generate visual plots

Linear Regression

Decision Tree

Test the model

Testing the effectiveness and results of the models

RMSE

Data Cleaning

Check missing value

Before

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
longitude             20640 non-null float64
latitude              20640 non-null float64
housing_median_age    20640 non-null float64
total_rooms           20640 non-null float64
total_bedrooms        20433 non-null float64
population            20640 non-null float64
households            20640 non-null float64
median_income         20640 non-null float64
median_house_value    20640 non-null float64
ocean_proximity       20640 non-null object
dtypes: float64(9), object(1)
memory usage: 1.6+ MB
```

Drop missing value

After

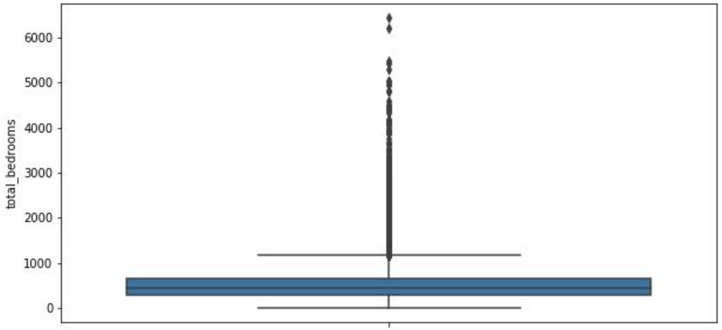
```
data_df = data.dropna(axis=0)
data_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 20433 entries, 0 to 20639
Data columns (total 10 columns):
longitude             20433 non-null float64
latitude              20433 non-null float64
housing_median_age    20433 non-null float64
total_rooms           20433 non-null float64
total_bedrooms        20433 non-null float64
population            20433 non-null float64
households            20433 non-null float64
median_income         20433 non-null float64
median_house_value    20433 non-null float64
ocean_proximity       20433 non-null object
dtypes: float64(9), object(1)
memory usage: 1.7+ MB
```

Remove Outliers

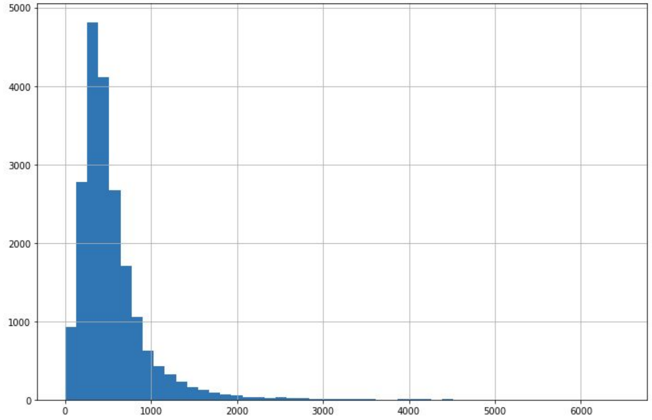
```
In [13]: plt.figure(figsize=(10,5))
         sns.boxplot(y='total_bedrooms',data=data_df)
         plt.plot

Out[13]: <function matplotlib.pyplot.plot(*args, scalex=True, scaley=True, data=None, **kwargs)>
```




```
In [12]: total_bedroms = data_df[data_df["total_bedrooms"].notnull()]["total_bedrooms"]
         total_bedroms.hist(figsize=(12,8),bins=50)

Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x124d31390>
```



Clean up Outliers

In [9]:
```
# Clean up outliners
def getOutliers(dataframe,column):
    column = "total_bedrooms"
    #housing[column].plot.box(figsize=(8,8))
    des = dataframe[column].describe()
    desPairs = {"count":0,"mean":1,"std":2,"min":3,"25":4,"50":5,"75":6,"max":7}
    Q1 = des[desPairs['25']]
    Q3 = des[desPairs['75']]
    IQR = Q3-Q1
    lowerBound = Q1-1.5*IQR
    upperBound = Q3+1.5*IQR
    print("(IQR = {})Outlier are anything outside this range: ({},{})".format(IQR,lowerBound,upperBound))
    #b = df[(df['a'] > 1) & (df['a'] < 5)]
    outliners = data_df[(data_df[column] < lowerBound) | (data_df[column] > upperBound)]

    print("Outliers out of total = {} are \n {}".format(data_df[column].size,len(outliners[column])))
    #remove the outliers from the dataframe
    outlierRemoved = data_df[~data_df[column].isin(outliners[column])]
    return outlierRemoved
```

Clean up Outliers

In [10]:
```
#get the outlier
data_df = getOutliers(data_df,"total_bedrooms")
data_df = getOutliers(data_df,"median_house_value")
data_df = getOutliers(data_df,"median_income")
data_df = getOutliers(data_df,"total_rooms")
data_df.info()
```

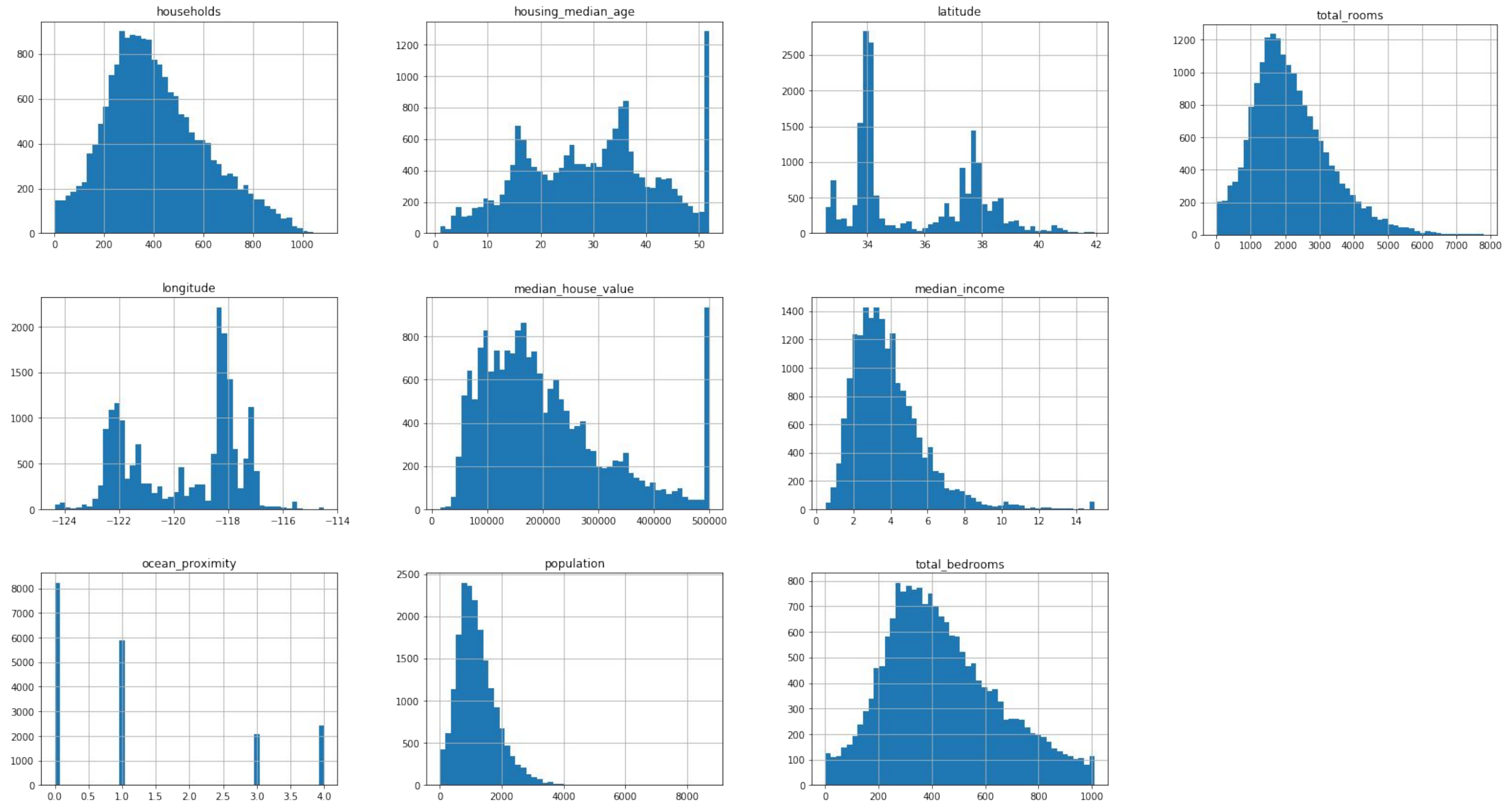
```
(IQR = 351.0)Outlier are anything outside this range: (-230.5,1173.5)
Outliers out of total = 20433 are
 1271
(IQR = 307.0)Outlier are anything outside this range: (-172.5,1055.5)
Outliers out of total = 19162 are
 365
(IQR = 294.0)Outlier are anything outside this range: (-155.0,1021.0)
Outliers out of total = 18797 are
 139
(IQR = 291.0)Outlier are anything outside this range: (-151.5,1012.5)
Outliers out of total = 18658 are
 42
<class 'pandas.core.frame.DataFrame'>
Int64Index: 18616 entries, 0 to 20639
Data columns (total 10 columns):
longitude             18616 non-null float64
latitude              18616 non-null float64
housing_median_age    18616 non-null float64
total_rooms           18616 non-null float64
total_bedrooms        18616 non-null float64
population            18616 non-null float64
households            18616 non-null float64
median_income         18616 non-null float64
median_house_value    18616 non-null float64
ocean_proximity       18616 non-null object
dtypes: float64(9), object(1)
memory usage: 1.6+ MB
```

Convert non-numerical value to numerical number

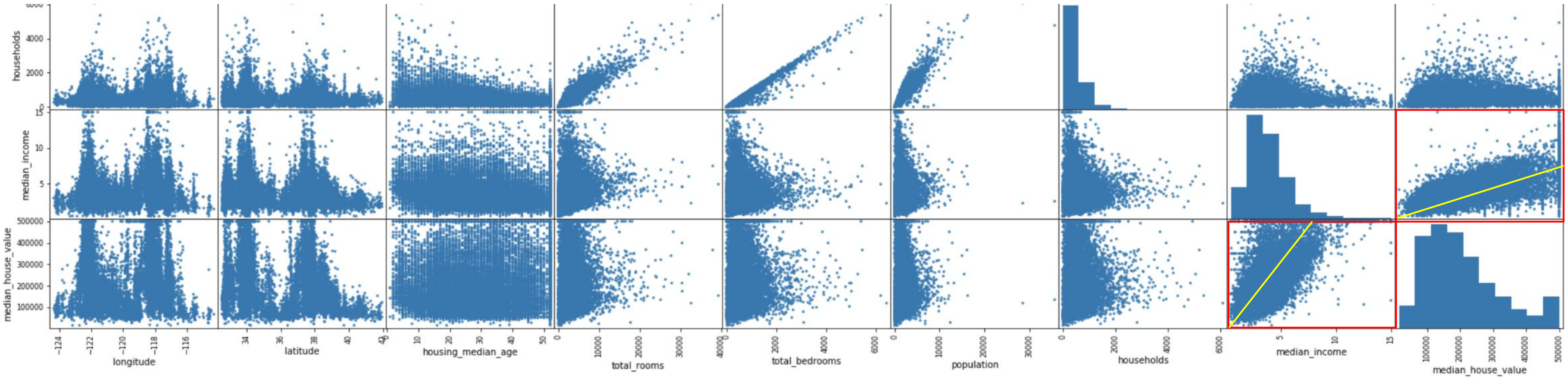
ocean_proximity
NEAR BAY
NEAR BAY
NEAR BAY
NEAR BAY
NEAR BAY
NEAR BAY
NEAR BAY
NEAR BAY
NEAR BAY
NEAR BAY
NEAR BAY

ocean_proximity
3
3
3
3
3
3
3
3
3
3
3

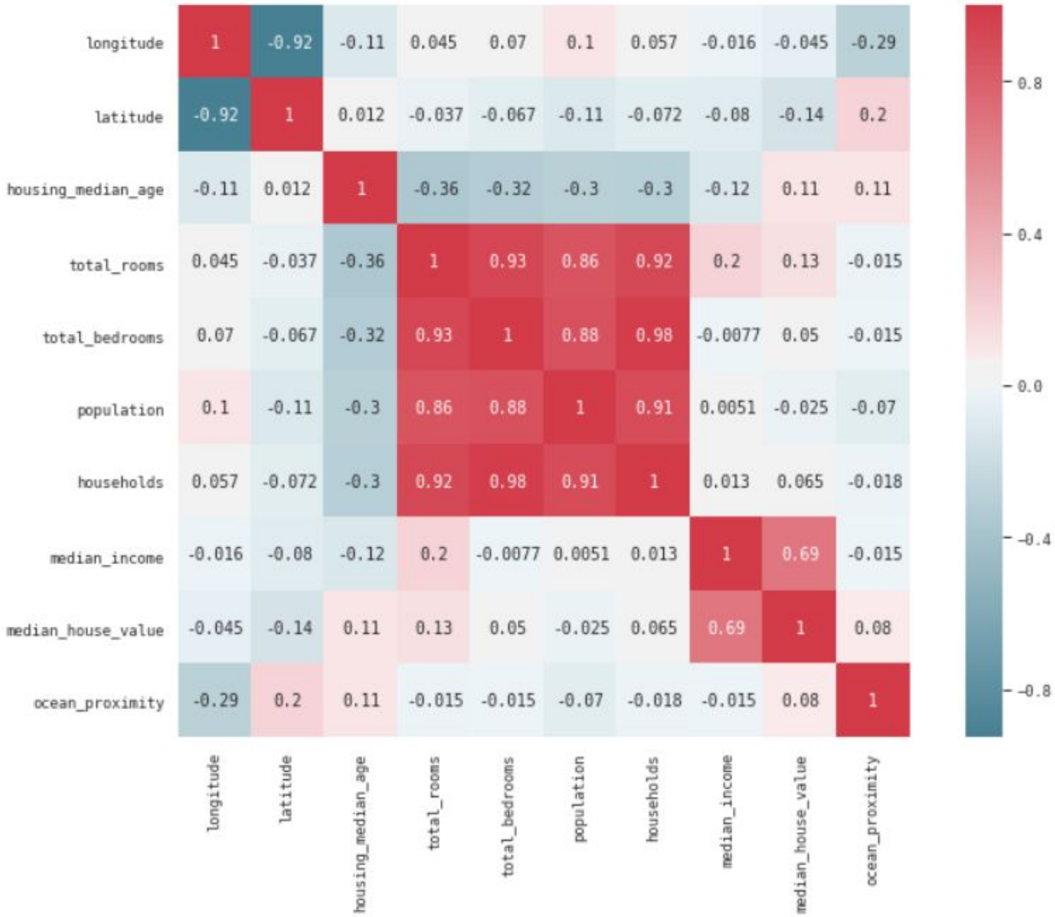
Histogram of each column



Scatter plot of correlation



Median_income & Median_house_value


longitude
latitude
housing_median_age
total_rooms
total_bedrooms
population
households
median_income
median_house_value
ocean_proximity
1
-0.92
-0.11
0.045
0.07
0.1
0.057
-0.016
-0.045
-0.29
-0.92
1
0.012
-0.037
-0.067
-0.11
-0.072
-0.08
-0.14
0.2
-0.11
0.012
1
-0.36
-0.32
-0.3
-0.3
-0.12
0.11
0.11
0.045
-0.037
-0.36
1
0.93
0.86
0.92
0.2
0.13
-0.015
0.07
-0.067
-0.32
0.93
1
0.88
0.98
-0.0077
0.05
-0.015
0.1
-0.11
-0.3
0.86
0.88
1
0.91
0.0051
-0.025
-0.07
0.057
-0.072
-0.3
0.92
0.98
0.91
1
0.013
0.065
-0.018
-0.016
-0.08
-0.12
0.2
-0.0077
0.0051
0.013
1
0.69
-0.015
-0.045
-0.14
0.11
0.13
0.05
-0.025
0.065
0.69
1
0.08
-0.29
0.2
0.11
-0.015
-0.015
-0.07
-0.018
-0.015
0.08
1
0.8
0.4
0.0
-0.4
-0.8

Apply Linear Regression & Decision Tree

Apply LinearRegression

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

# X_train = data_df.drop(['total_rooms','total_bedrooms','households',
#                        'ocean_proximity','median_house_value','longitude','latitude','housing_median_age'
X_train = data_df.drop(['median_house_value'],axis=1)
Y_train = data_df['median_house_value']

X,X_test,Y,Y_test = train_test_split(X_train, Y_train, test_size=0.2)
```

```
clf = LinearRegression()
clf.fit(np.array(X),Y)
confidence = clf.score(X, Y)
print("confidence: ", confidence)
```

```
confidence:  0.6537004439295262
```

```
#initantiate the linear regression
linearRegModel = LinearRegression(n_jobs=-1)
#fit the model to the training data (learn the coefficients)
linearRegModel.fit(X_train,Y_train)
#print the intercept and coefficients
print("Intercept is "+str(linearRegModel.intercept_))
print("coefficients  is "+str(linearRegModel.coef_))
```

```
Intercept is -3544080.1333994106
coefficients  is [-4.22553578e+04 -4.23296088e+04  1.12187140e+03 -1.50175843e+01
  2.58484960e+02 -5.83804855e+01  3.37103493e+00  4.15396026e+04
 -8.71153033e+02]
```

```
y_pred = linearRegModel.predict(X_test)
```

```
test = pd.DataFrame({'Predicted':y_pred,'Actual':Y_test})
fig= plt.figure(figsize=(16,8))
test = test.reset_index()
test = test.drop(['index'],axis=1)
plt.plot(test[:50])
plt.legend(['Actual','Predicted'])
sns.jointplot(x='Actual',y='Predicted',data=test,kind='reg',);
```

Perform Decision Tree Regression

```
from sklearn.tree import DecisionTreeRegressor
dtReg = DecisionTreeRegressor(max_depth=10)
dtReg.fit(X_train,Y_train)
confidence = dtReg.score(X, Y)
print("confidence: ", confidence)
```

```
confidence:  0.8426651459087362
```

```
dtReg_y_pred = dtReg.predict(X_test)
dtReg_y_pred
```

```
array([111175.75757576, 321063.63636364, 146022.07792208, ...,
       495529.28571429, 196902.77777778, 129352.94117647])
```

```
print(len(dtReg_y_pred))
print(len(Y_test))
print(dtReg_y_pred[0:5])
print(Y_test[0:5])
```

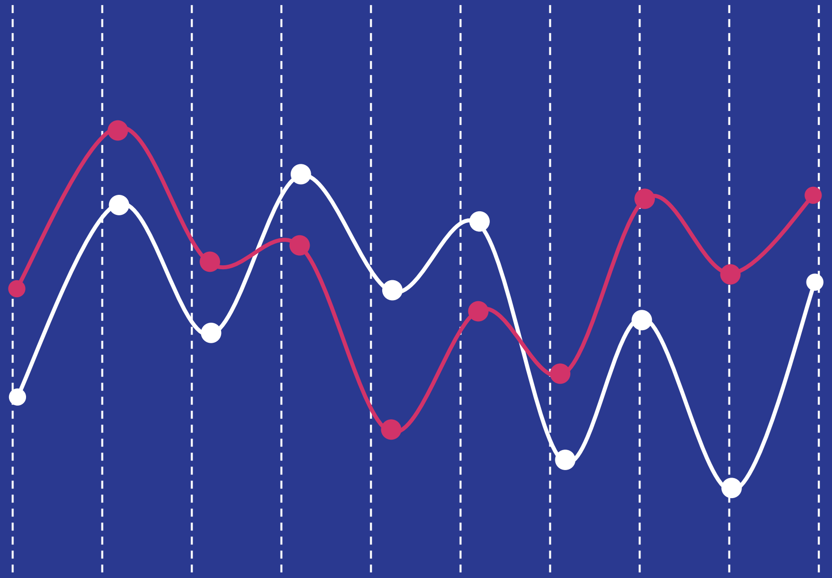
```
3724
3724
[111175.75757576 321063.63636364 146022.07792208  81260.66666667
  84745.18072289]
9943      106700.0
4707      250000.0
8410      170600.0
13983      57400.0
13580      77900.0
Name: median_house_value, dtype: float64
```

```
print(np.sqrt(metrics.mean_squared_error(Y_test,dtReg_y_pred)))
```

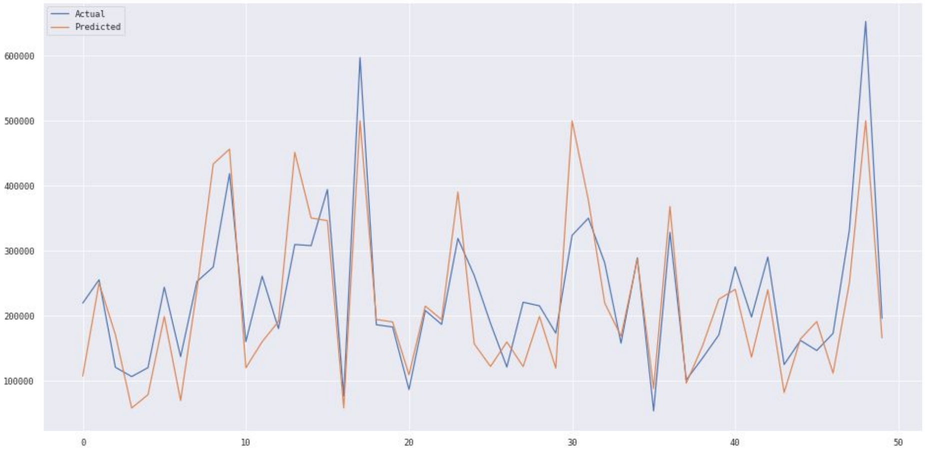
```
46049.44911738237
```

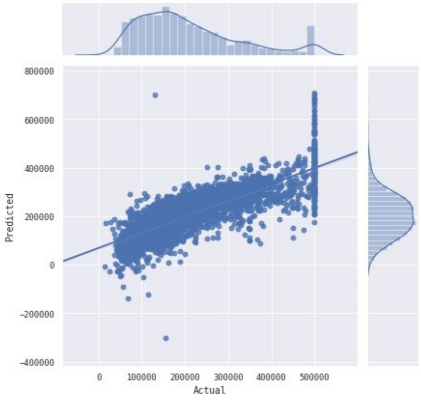
```
test = pd.DataFrame({'Predicted':dtReg_y_pred,'Actual':Y_test})
fig= plt.figure(figsize=(16,8))
test = test.reset_index()
test = test.drop(['index'],axis=1)
plt.plot(test[:50])
plt.legend(['Actual','Predicted'])
sns.jointplot(x='Actual',y='Predicted',data=test,kind="reg")
```

Testing and Results

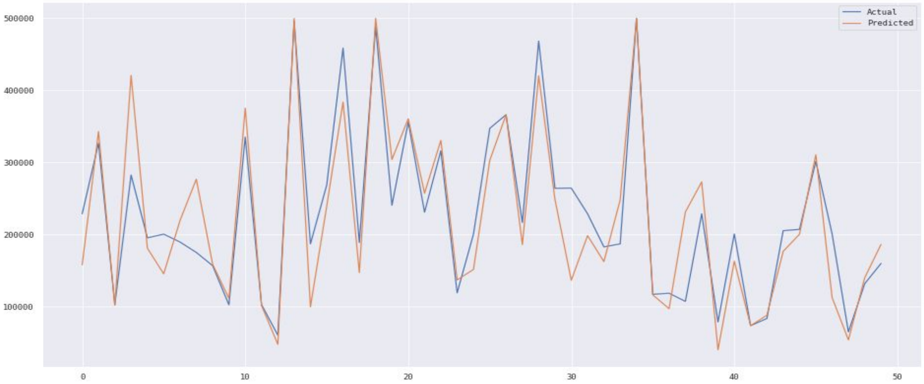


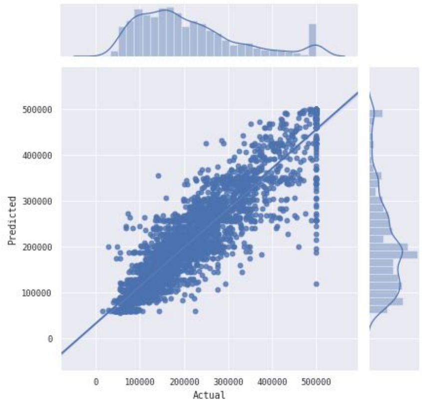
Predicted vs. Actual





Linear Regression
RMSE = 68541.16
Accuracy = 0.65





Decision Tree
max_depth = 10
RMSE = 46049.45
Accuracy = 0.84

Solution

The linear correlation between median_house_value and median income is obvious.

Decision Tree model has a better performance than Linear Regression model in this case.