Comparison of different regression methods for house price prediction

Viji Venugopal

Abstract

Buying a house is a big decision in one's life and it involves huge investment. Being able to predict the price of a house given its features, helps one in making the right decision. The objective of this project is to predict house prices in King County, USA using regression models and to identify the best fitting model. Three regression models were used in the study: Multiple Linear Regression, Decision Tree Regression and Random Forest Regression. The dataset for this study is obtained from Kaggle (https://www.kaggle.com/harlfoxem/housesalesprediction). The best regression model is identified by comparing the r2score for the different models.

Motivation

This project aims at predicting the house prices, given its features. Knowing the indicators which decide the house price will enable one to look for the right features and make the decision to purchase the house at the best rate, thereby making it a good investment for the future. This also helps the seller to estimate the selling cost of a housing property.

Dataset(s)

The dataset for this study was obtained from Kaggle:
(https://www.kaggle.com/harlfoxem/housesalesprediction)

This dataset contains house sale prices for King County, USA. It includes homes sold between May 2014 and May 2015. The dataset has 21613 rows of house sales data with 21 variables, which serves as features of the dataset, were then used to predict sales price.

Data Preparation and Cleaning

The dataset contained 21613 rows of house sales data with 21 variables representing housing prices traded between May 2014 and May 2015. The following data cleaning and preprocessing were employed.

- Check for missing data – There weren’t any.
- Remove ambiguous data – It appeared illogical to have a 33-bedroom single story house on 6000 sq.ft lot and only 1.75 bathrooms. This row was removed.
- The age of the house was calculated by deducting the year the house was built / renovated from the year the data was compiled (2015) to study the dependency of house price on aging.
- Price bins were added to data column to assess the price variation with geographic location.

Research Questions

1. Identify a regression model which can give us a good prediction on the price of the house based on other variables.

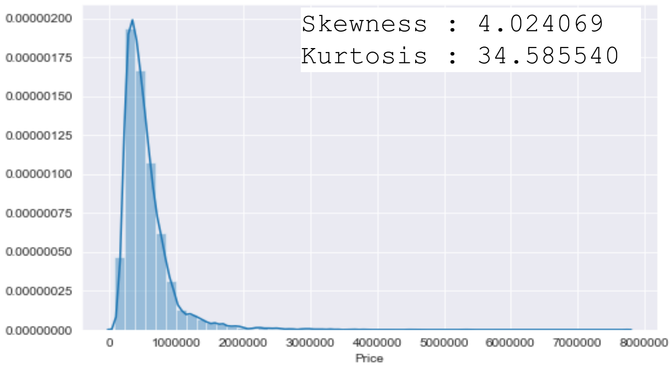
2. Which location in King County is best to invest?

3. What features should a builder look for when planning a new project in King County?

Methods

Exploratory data analysis was done to identify patterns and relationships in the data which may help subsequent analysis and house price prediction.

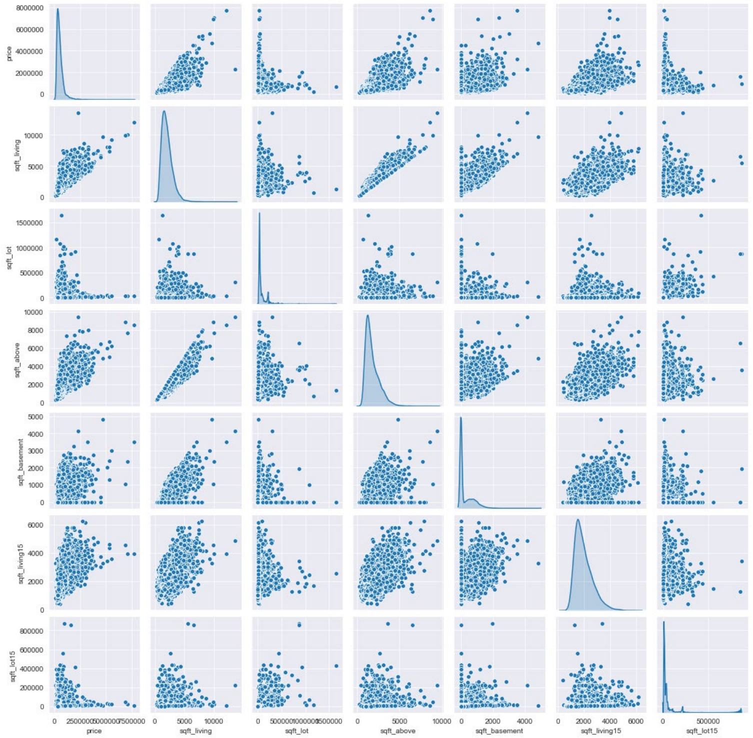
Distribution of target variable : sales price



Presumption:

- Positive skewed distribution curve may be an indication that many houses are sold at less than the average value.
- High Kurtosis may be because of outliers present in the data.

Pairplot showing relationship between continuous variables

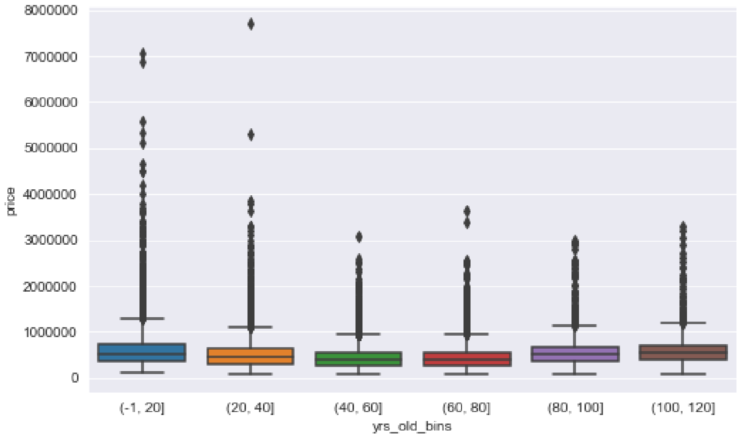


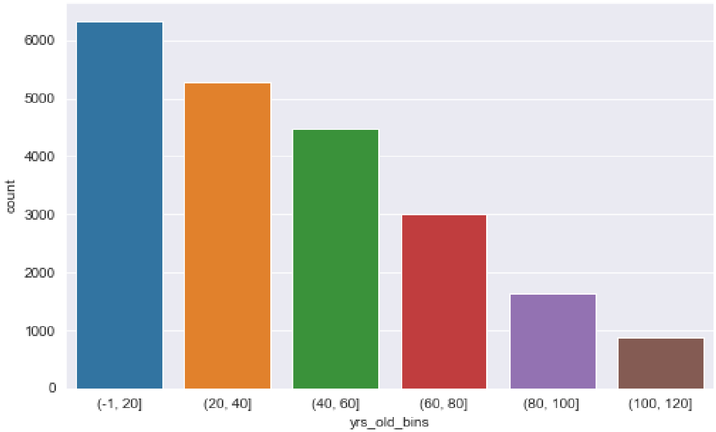
- The diagonal plots show the distribution of the continuous variables. Almost all the variables are positively skewed.
- The non-diagonal plots show the relationship between pairs of continuous variables. Variables 'sqft_above' and 'sqft_living' are almost linearly related. ‘price’ is also almost linearly related to 'sqft_living’.

Relationship between ordinal variables and house price

- Bedrooms & Bathrooms:The median house price is going up with increase in the number of bedrooms (upto 7) and bathrooms (upto 5). Thereafter it doesn't show a linear trend.
- Floors: The median house price increases with an increase in the number of floors (upto 2.5)
- Waterfront: The houses with waterfront are priced higher.
- View: The better the view, the higher the price.
- Condition: The median price for condition 3, 4 and 5 remains almost the same, though price for condition 1 & 2 houses are slightly lower.
- Grade: The median house price increases almost exponentially with increase in grade.

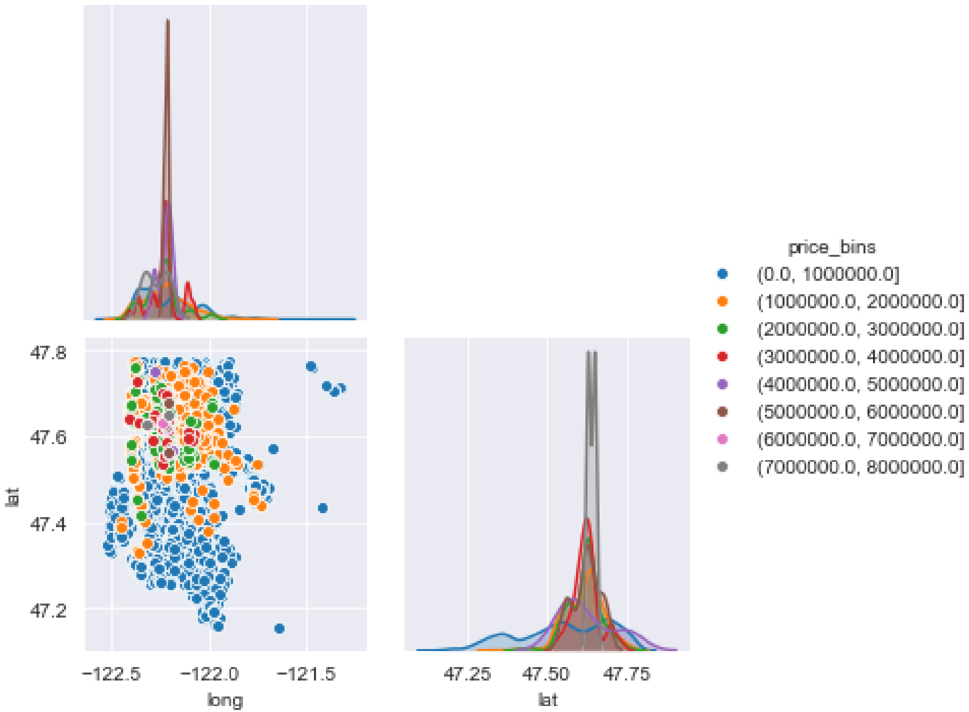
Relationship between age of the house and house price





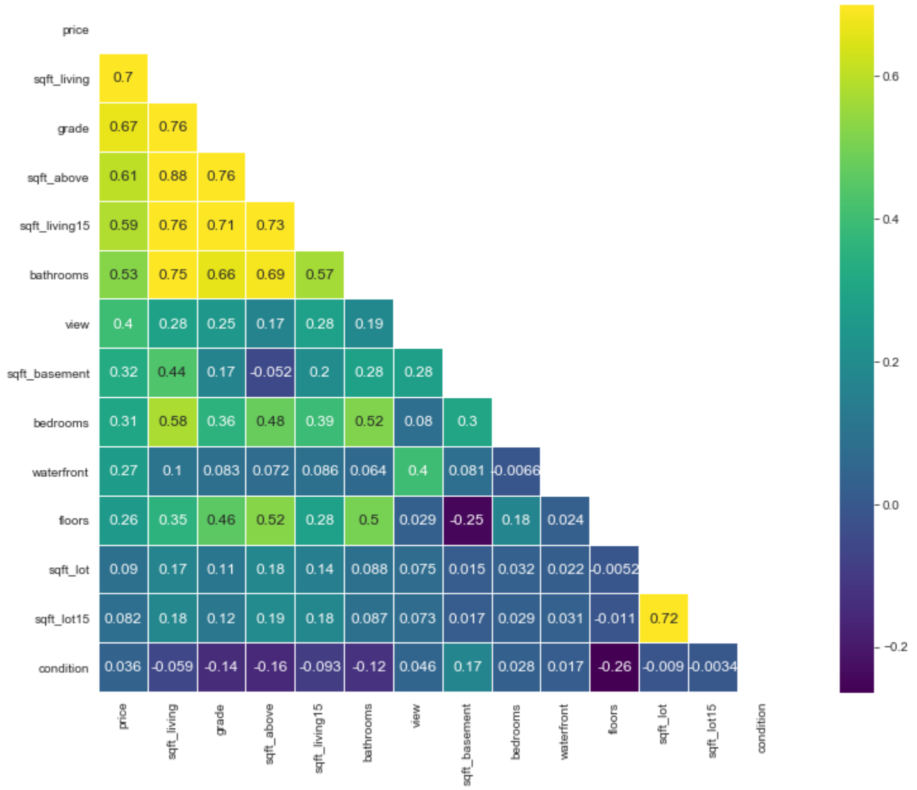
- There is not much change in the median house price with aging.
- Therefore, age factor was discarded in further analysis

Relationship between Geographical location and house price



- Higher priced houses are located in some specific regions. Specifically, between latitudes of 47.5° and 47.7° and longitudes of −122.0° and −122.4°.
- Geographical location (latitude, longitude) is a key factor that decides house price.

Correlation between variables



- Correlation between variables in the dataset is computed and are arranged in the order of their correlation with target variable, price.
- Yellow squares represent the most correlated variables.

Feature selection

- Variables which are highly correlated with target variable, price are identified from the correlation matrix.
- The varaibles highly correlated with price are - 'sqft_living', 'grade', 'sqft_above', 'sqft_living15', 'bathrooms', 'view', 'sqft_basement', 'bedrooms', 'waterfront', 'floors’.
- 'sqft_living' and 'sqft_above' are highly correlated to each other. Therefore, only 'sqft_living’ is included in the training feature as it has a higher correlation with ‘price’ than 'sqft_above’.
- Exploratory data analysis showed that geographical location (latitude, longitude) is a key factor that decides house price. Hence these features are also included as training features.

Findings:

Model Selection

1. Multiple Linear Regression

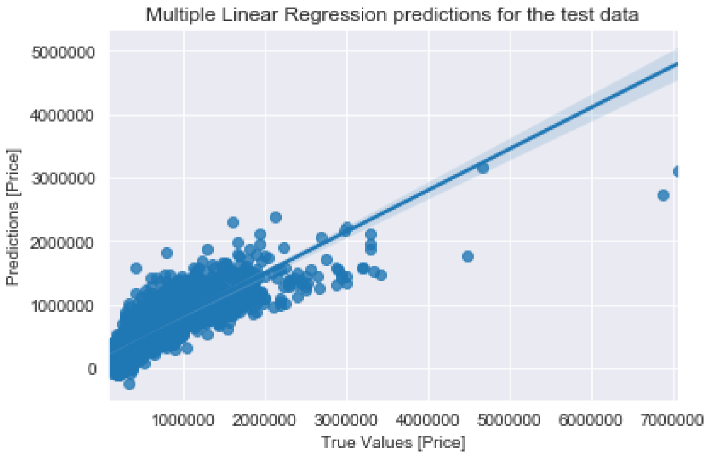
Multiple linear regression (MLR) attempts to model a linear relationship between the several explanatory (independent) variables and the response (dependent) variable. The dataset is split in the ratio 80:20 for training and testing. After that, the model was trained and then used it to run predictions.

Model evaluation

RMSE: 208974.066

MAE: 133365.201

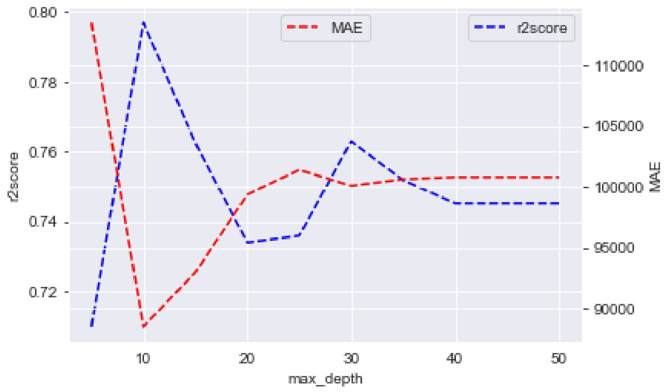
R2score: 0.676



2. Decision Tree Regression

Decision tree regression divides the data into multiple splits that typically answer a simple if-else condition. In this model, the DecisionTreeRegressor class provided by sklearn is used. DecisionTreeRegressor has a max_depth parameter that controls the size of the tree. Hyperparameter tuning is done to identify the best value for max_depth.

max_depth	RMSE	MAE	r2score
5	197738.11	113512.93	0.71
10	165407.15	88514.64	0.80
15	178910.89	92964.29	0.76
20	189354.61	99370.01	0.73
25	188644.19	101403.71	0.74
30	178732.86	100080.28	0.76
35	182904.86	100619.37	0.75
40	185296.70	100791.27	0.75
45	185296.70	100791.27	0.75
50	185296.70	100791.27	0.75



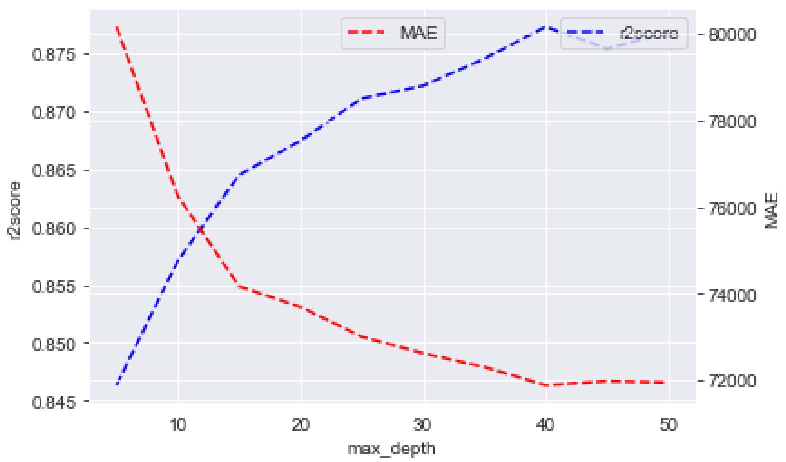
Best fitting Decision Tree Model

DecisionTreeRegressor(*, criterion='mse', splitter='best', max_depth= 10, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state= 100, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, presort='deprecated', ccp_alpha=0.0)

3. Random Forest Regression

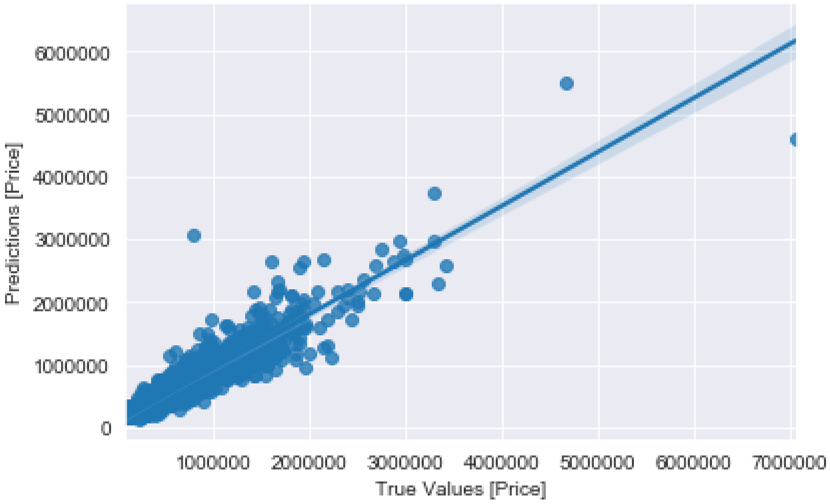
Random forest is a supervised learning algorithm which uses ensemble learning method by constructing a multitude of decision trees at training time and outputting the mean/average prediction of the individual trees. In this model, the RandomForestRegressor class provided by sklearn is used. RandomForestRegressor has n_estimators parameter that controls the number of trees in the forest. Hyperparameter tuning is done to identify the best value for n_estimators.

n_estimators	RMSE	MAE	r2score
5	143920.145	80161.515	0.846
10	138789.519	76253.773	0.857
15	135135.123	74162.498	0.864
20	133648.282	73687.905	0.867
25	131788.472	72999.800	0.871
30	131226.704	72619.218	0.872
35	130019.491	72299.711	0.875
40	128573.022	71879.346	0.877
45	129563.984	71974.844	0.875
50	128841.409	71944.298	0.877



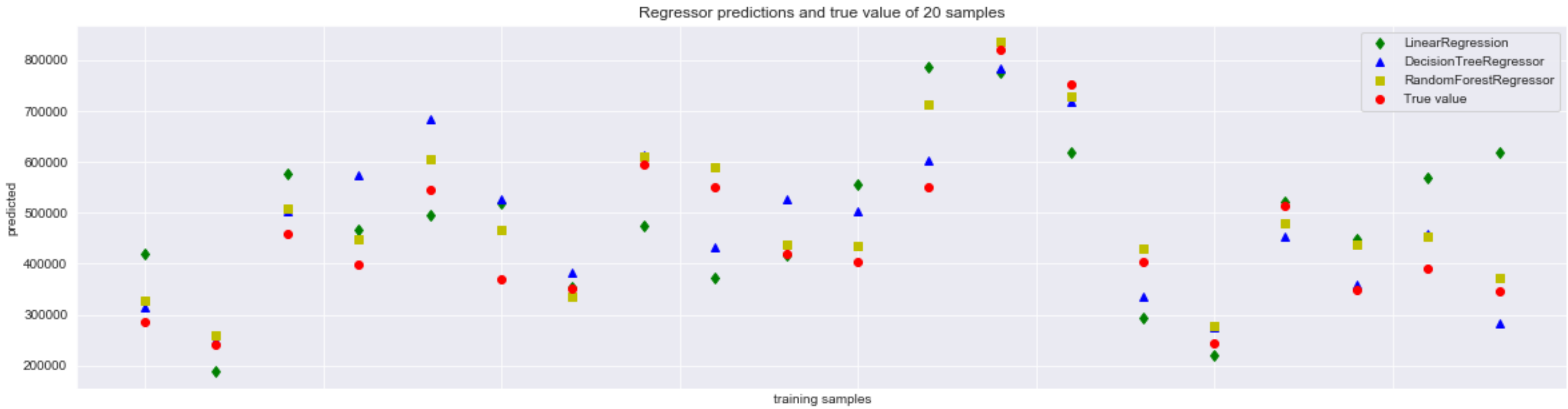
Best fitting Random Forest Regression Model

RandomForestRegressor(n_estimators= 40, *, criterion='mse', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, bootstrap=True, oob_score=False, n_jobs=None, random_state= 0, verbose=0, warm_start=False, ccp_alpha=0.0, max_samples=None)



Comparing the different regressor models

Predictive model	Multiple Linear Regression	Decision Tree Regression	Random Forest Regression
RMSE	212258.51	158430.57	129681.01
MAE	133604.59	88742.27	72253.66
r2score	0.66	0.81	0.87



The highest accuracy (r2score = 0.87) is obtained with Random Forest Regression model.

Limitations

- Only three regression models were analyzed in this study. There are other regression algorithms like XGBoost, LightGBM etc. which may give better accuracy. Also, further research can be conducted to investigate how the combinations of different models work.

- Cross-validation was not done to check for overfitting.

- Among the compared models, Random Forest method has the highest accuracy, but its run-time is high since the dataset must be fit multiple times.

Conclusions

1. Among the models studied in this work, Random Forest regression model gives highest accuracy in house price prediction.
2. High valued houses are located between latitudes of 47.5° and 47.7° and longitudes of -122.0° and -122.4°. This may be an ideal location to invest in King County. However, the investment decision should be based on one's financial provisions and aspirations.
3. The most sold out homes are either single or two storied houses with 3 or 4 bedrooms. Better view and having waterfront raises the value of the house. It would be good for a builder to keep these in mind while planning a new project.

Acknowledgements

The dataset for this study is obtained from Kaggle
(https://www.kaggle.com/harlfoxem/housesalesprediction)

References

1. House Sales in King County, USA Dataset : https://www.kaggle.com/harlfoxem/housesalesprediction

2. https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html

3. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

4. https://seaborn.pydata.org/

DSE 200X_Comparison of Regression methods for House price prediction

Predict house price using regression

This dataset https://www.kaggle.com/harlfoxem/housesalesprediction (https://www.kaggle.com/harlfoxem/housesalesprediction) is taken from Kaggle and contains house sale prices for King County. It includes homes sold between May 2014 and May 2015.

Import libraries

In [1]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score
from math import sqrt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
```

Read input files

In [2]:

```
df = pd.read_csv('./Downloads/kc_house_data/kc_house_data.csv')
df.head()
```

Out[2]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors
0	7129300520	20141013T000000	221900.0	3	1.00	1180	5650	1.0
1	6414100192	20141209T000000	538000.0	3	2.25	2570	7242	2.0
2	5631500400	20150225T000000	180000.0	2	1.00	770	10000	1.0
3	2487200875	20141209T000000	604000.0	4	3.00	1960	5000	1.0
4	1954400510	20150218T000000	510000.0	3	2.00	1680	8080	1.0

5 rows × 21 columns

Data Exploration

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21613 entries, 0 to 21612
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   id             21613 non-null  int64  
 1   date           21613 non-null  object 
 2   price          21613 non-null  float64
 3   bedrooms       21613 non-null  int64  
 4   bathrooms      21613 non-null  float64
 5   sqft_living    21613 non-null  int64  
 6   sqft_lot       21613 non-null  int64  
 7   floors         21613 non-null  float64
 8   waterfront     21613 non-null  int64  
 9   view           21613 non-null  int64  
 10  condition      21613 non-null  int64  
 11  grade          21613 non-null  int64  
 12  sqft_above     21613 non-null  int64  
 13  sqft_basement  21613 non-null  int64  
 14  yr_built       21613 non-null  int64  
 15  yr_renovated   21613 non-null  int64  
 16  zipcode        21613 non-null  int64  
 17  lat            21613 non-null  float64
 18  long           21613 non-null  float64
 19  sqft_living15  21613 non-null  int64  
 20  sqft_lot15     21613 non-null  int64  
dtypes: float64(5), int64(15), object(1)
memory usage: 3.5+ MB
```

The dataset has 21613 rows of house sales data and 21 columns. There is no missing value in the dataset.

In [4]:
```
df.describe()
```

Out[4]:

	id	price	bedrooms	bathrooms	sqft_living	sqft_lot
count	2.161300e+04	2.161300e+04	21613.000000	21613.000000	21613.000000	2.161300e+04
mean	4.580302e+09	5.400881e+05	3.370842	2.114757	2079.899736	1.510697e+04
std	2.876566e+09	3.671272e+05	0.930062	0.770163	918.440897	4.142051e+04
min	1.000102e+06	7.500000e+04	0.000000	0.000000	290.000000	5.200000e+02
25%	2.123049e+09	3.219500e+05	3.000000	1.750000	1427.000000	5.040000e+03
50%	3.904930e+09	4.500000e+05	3.000000	2.250000	1910.000000	7.618000e+03
75%	7.308900e+09	6.450000e+05	4.000000	2.500000	2550.000000	1.068800e+04
max	9.900000e+09	7.700000e+06	33.000000	8.000000	13540.000000	1.651359e+06

In [5]:
```
#Find the number of unique entries in each column
df.nunique()
```

Out[5]:
```
id               21436
date               372
price             4028
bedrooms            13
bathrooms           30
sqft_living       1038
sqft_lot          9782
floors               6
waterfront           2
view                 5
condition            5
grade               12
sqft_above         946
sqft_basement      306
yr_built           116
yr_renovated        70
zipcode             70
lat               5034
long               752
sqft_living15      777
sqft_lot15        8689
dtype: int64
```

Classify the variables into 4 categories.

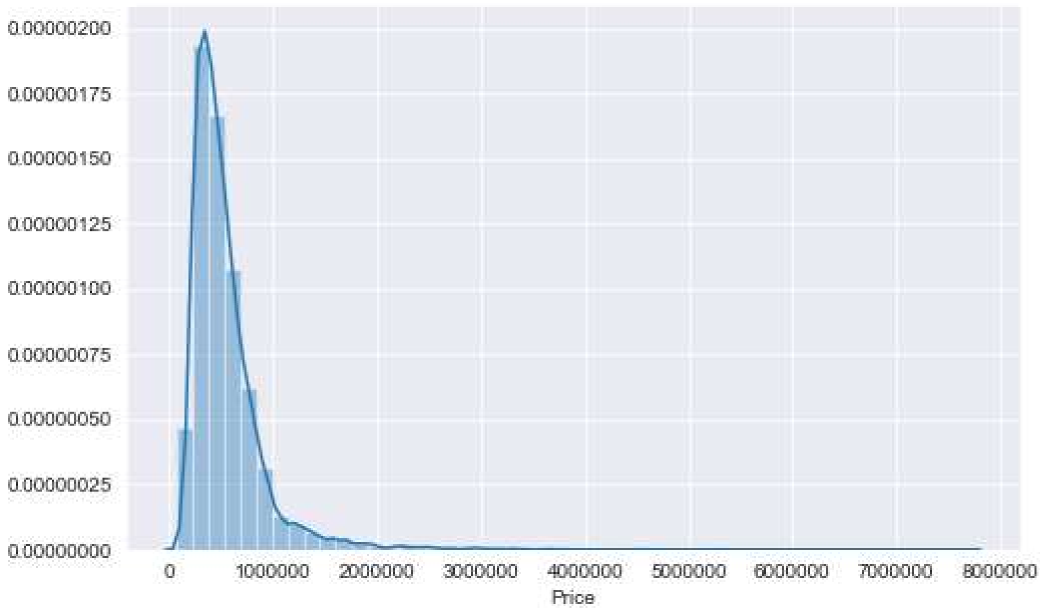
- Continuous variables: A numeric variable that takes any value between a certain set of real numbers.
- Discrete variables: A numeric variable that can only take distinct and separate values.
- Nominal variables:A categorical variable which has no order.
- Ordinal variables: A categorical variable whose value can be logically ordered or ranked.

In [6]:
```
continuous_variables = ['price', 'sqft_living', 'sqft_lot', 'sqft_above', 'sq
discrete_variables = ['yr_built', 'yr_renovated']
nominal_variables = ['lat', 'long', 'zipcode']
ordinal_variables = ['bedrooms', 'bathrooms', 'floors', 'waterfront', 'view',
```

Distribution of target variable : sales price

In [7]:
```
sns.set_style("darkgrid")
plt.figure(figsize =(8,5))
sns.distplot(df['price'], axlabel = 'Price')
```

Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0x1fdcf2bda88>



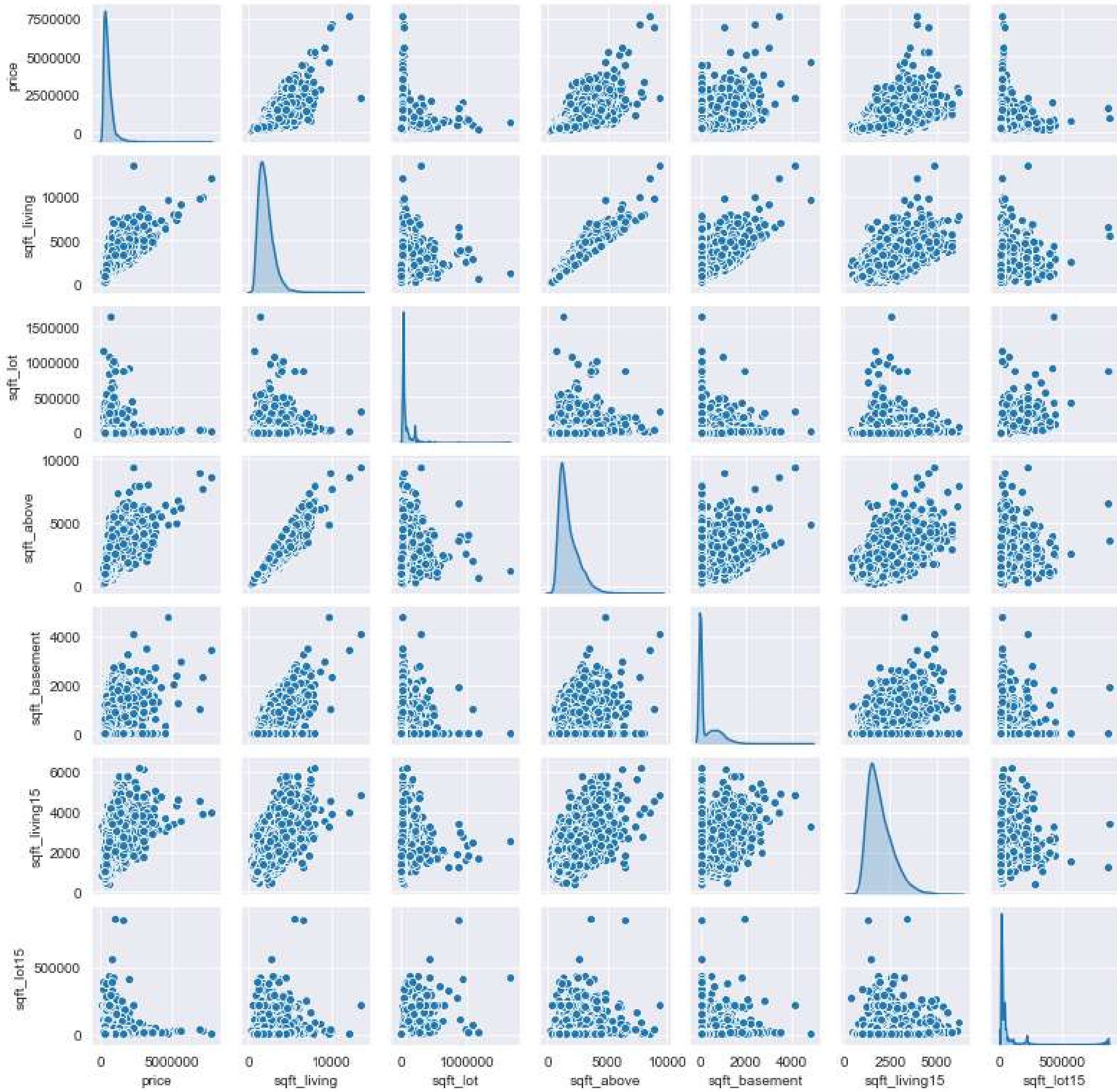
In [8]:
```
print('Skewness : %f' % df['price'].skew())
print('Kurtosis : %f' % df['price'].kurt())
```

```
Skewness : 4.024069
Kurtosis : 34.585540
```

Skewness is the degree of distortion from the symmetrical bell curve or the normal distribution. The above distribution curve shows a positive skewness. ie, the peak of the distribution curve is less than the average value. This may be an indication that many houses are sold at less than the average value. Kurtosis is a measure of whether the data are heavy-tailed or light-tailed relative to a normal distribution. High Kurtosis (34.585540) in this case may be because of outliers present in the data.

In [9]:
```
sns.pairplot(df[continuous_variables], height = 1.5 ,kind ='scatter', diag_ki
```

Out[9]: <seaborn.axisgrid.PairGrid at 0x1fdcfd8ff88>



Almost all the continuous variables show a positive skewness. Variables 'sqft_above' and 'sqft_living' are almost linearly related.

In [10]:

```
fig, ax = plt.subplots(7, 3, figsize=(15,20))

for i, el in enumerate(ordinal_variables):
    feature_count = df[el].value_counts()
    sns.set_style("darkgrid")
    sns.countplot(x=el, data=df,  ax=ax[i,0])
    sns.boxplot(x=el, y= 'price',data=df, ax=ax[i,1])
    sns.regplot(x=el, y= 'price',data=df,  ax=ax[i,2])

plt.show()
```

Observations

- Bedrooms & Bathrooms:The median house price is going up with increase in the number of bedrooms (upto 7) and bathrooms (upto 5). Thereafter it doesn't show a linear trend.
- Floors: The median house price increases with an increase in the number of floors (upto 2.5)
- Waterfront: The houses with waterfront are priced higher.
- View: The better the view, the higher the price.
- Condition: The median price for condition 3, 4 and 5 remains almost the same, though price for condition 1 & 2 houses are slightly lower.
- Grade: The median house price increases almost exponentially with increase in grade.

House age vs. house price.

In [11]:

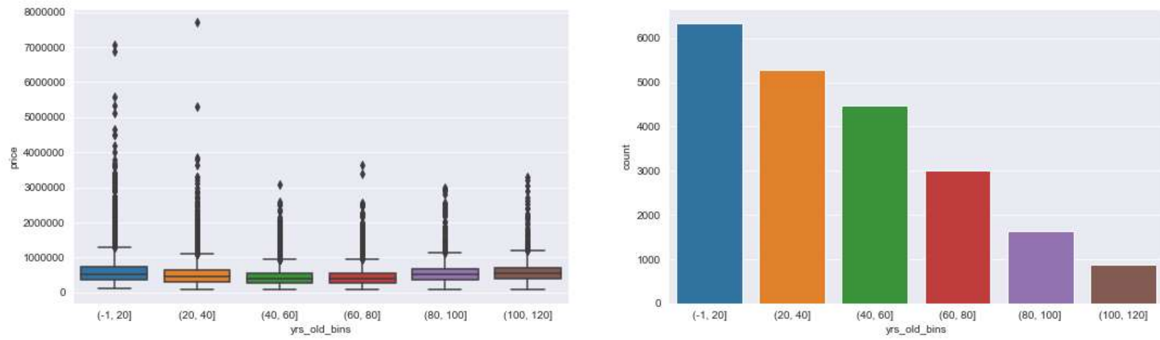
```
df1 = df.copy()
df1.drop(['id','date'], axis = 1, inplace=True)
df1['yrs_old_renovated'] = np.where(df1['yr_renovated']!= 0, 2015 - df1['yr_r
df1['yrs_old_bins'] = pd.cut(x = df1['yrs_old_renovated'], bins = [-1, 20, 40
df1['price_bins'] = pd.cut(x = df1['price'], bins = [0, 1e6, 2e6, 3e6, 4e6, 5
df1.head()
```

Out[11]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	g
0	221900.0	3	1.00	1180	5650	1.0	0	0	3	
1	538000.0	3	2.25	2570	7242	2.0	0	0	3	
2	180000.0	2	1.00	770	10000	1.0	0	0	3	
3	604000.0	4	3.00	1960	5000	1.0	0	0	5	
4	510000.0	3	2.00	1680	8080	1.0	0	0	3	

5 rows × 22 columns

In [12]:
```
fig, ax = plt.subplots(ncols=2, figsize=(18,5))
sns.boxplot(x='yrs_old_bins', y= 'price', data=df1, ax=ax[0])
sns.countplot(x='yrs_old_bins', data=df1, ax=ax[1])
plt.show()
```



There is not much change in the median house price with aging. So we will discard yr_built and yr_renovated features from the training data.

Let's have a look at the 33 bedroom house and compare it with mean and median values of the dataset.

In [13]:
```
df[(df['bedrooms'] == 33)]
```

Out[13]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	fl
15870	2402100895	20140625T000000	640000.0	33	1.75	1620	6000	

1 rows × 21 columns

It looks like there is some error in the data as it is illogical to have a 33 bedroom single story house on 6000 sqft_lot and with only 1.75 bathrooms. So I'm dropping this row.

In [14]:
```
df1.drop(df1[df1['bedrooms'] == 33].index, axis = 0, inplace = True)
```

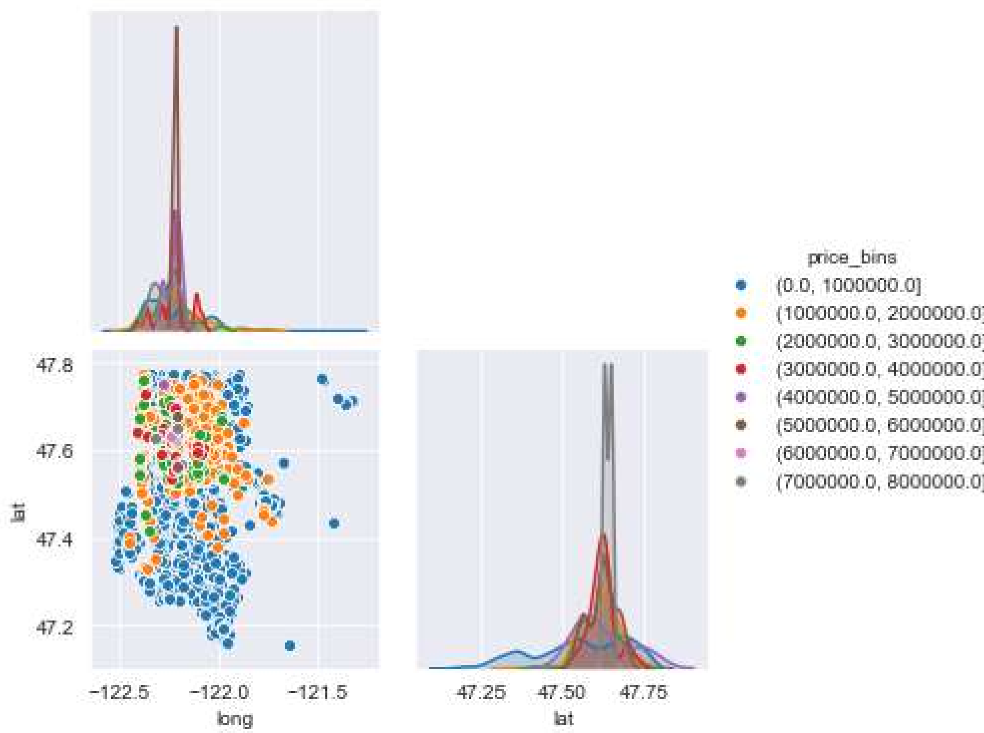
Geographical location vs. house price

In [15]:

```
plt.figure(figsize=(20,15))
g = sns.pairplot(data=df1[['long','lat','price_bins']], hue='price_bins', cor
```

```
C:\Users\vijis\anaconda3\lib\site-packages\seaborn\distributions.py:288: Us
erWarning: Data must have variance to compute a kernel density estimate.
  warnings.warn(msg, UserWarning)
C:\Users\vijis\anaconda3\lib\site-packages\seaborn\distributions.py:288: Us
erWarning: Data must have variance to compute a kernel density estimate.
  warnings.warn(msg, UserWarning)
```

```
<Figure size 1440x1080 with 0 Axes>
```

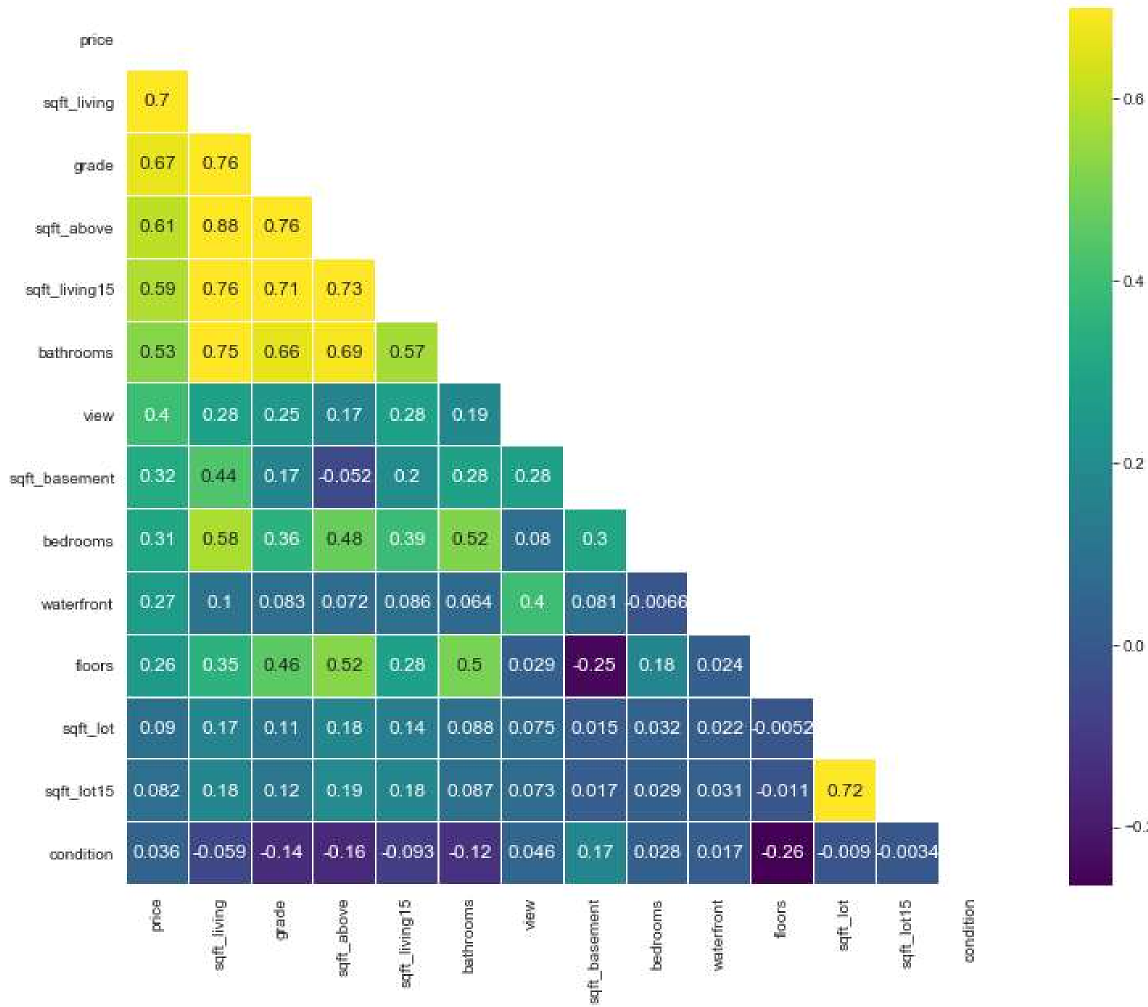


The above scatter plot is almost the shape of King County. It can be seen that higher priced houses are located in some specific regions, especially near the coasts. Specifically, the high priced houses are located between latitudes of 47.5^o and 47.7^o and longitudes of -122.0^o and -122.4^o. This information may be helpful for a homebuyer when making a purchase decision. This also indicates that geographical location (latitude, longitude) is a key factor that decides house price.

Correlation between variables

In [16]:

```
features = continuous_variables +  ordinal_variables
k= len(features)
cols = df1[features].corr().nlargest(k,'price')['price'].index
cm = np.corrcoef(df[cols].values.T)
mask = np.zeros_like(df1[cols].corr())
mask[np.triu_indices_from(mask)] = True
with sns.axes_style("white"):
    f, ax = plt.subplots(figsize=(15, 10))
    ax = sns.heatmap(cm, cmap='viridis', mask=mask, vmax=.7, linewidths=0.01,
                    linecolor="white",xticklabels = cols.values ,annot_kws =
```



Feature selection

Here we select the variables which are highly correlated with our target variable, price. Let's choose the top 10 varaibles - 'sqft_living', 'grade', 'sqft_above', 'sqft_living15', 'bathrooms', 'view', 'sqft_basement', 'bedrooms', 'waterfront', 'floors'.

'sqft_living' and 'sqft_above' are highly correlated with a correlation coefficient of 0.88. So keeping one of this variable in the training set is sufficient. 'sqft_living' has a higher correlation with 'price' than 'sqft_above'. Therefore, we will keep 'sqft_living' in the training feature. Also, we will add the geographical location parameters, 'lat' and 'long' in the training features.

In [17]:
```
selected_features = ['sqft_living', 'grade', 'sqft_living15', 'bathrooms', 'v
                     'waterfront', 'floors', 'long', 'lat']
target = ['price']
X = df[selected_features]
y = np.ravel(df[target])
```

In [18]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, rar
```

1. Multiple Linear Regression

Multiple linear regression (MLR) attempts to model a linear relationship between the several explanatory (independent) variables and the response (dependent) variable. Here we use all the selected independent training variables to predict the house price.

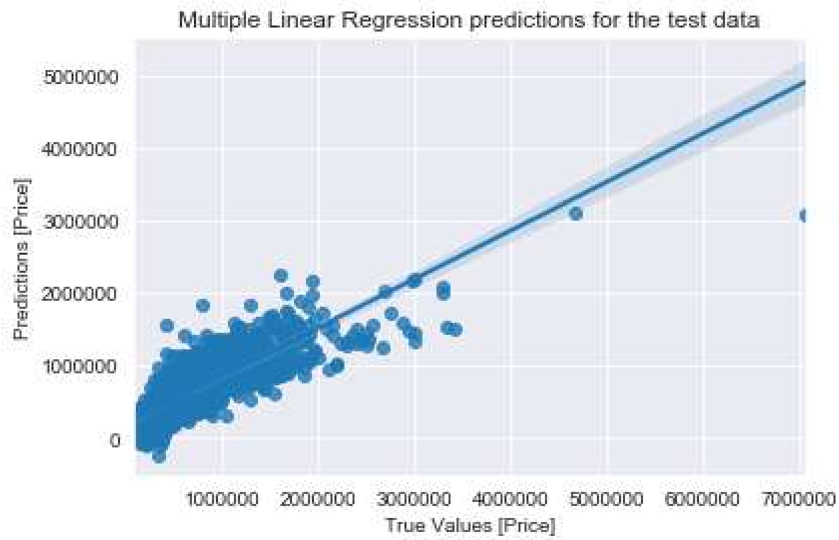
In [19]:
```
regressor = LinearRegression()
regressor.fit(X_train, y_train)
y_prediction = regressor.predict(X_test)
RMSE = sqrt(mean_squared_error(y_true = y_test, y_pred = y_prediction))
mae = mean_absolute_error(y_test, y_prediction)
r2score = r2_score(y_test, y_prediction)

print('RMSE:', RMSE)
print('MAE:' ,mae )
print('R2score:', r2score)
```

```
RMSE: 208974.06610008978
MAE: 133365.20097731653
R2score: 0.675963115586444
```

True Value vs. Predicted value for Multiple Linear Regression model

In [20]:
```
sns.regplot(x=y_test, y=  y_prediction)
plt.xlabel('True Values [Price]')
plt.ylabel('Predictions [Price]')
plt.title('Multiple Linear Regression predictions for the test data')
```

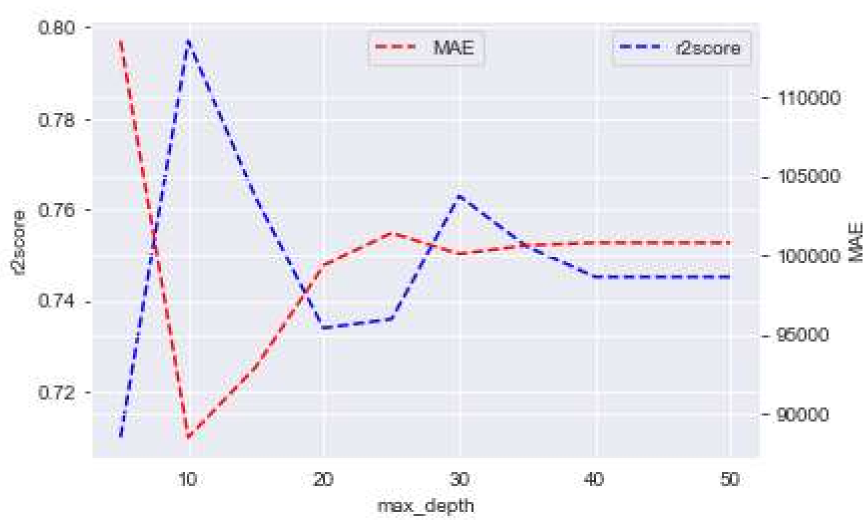
Out[20]:
```
Text(0.5, 1.0, 'Multiple Linear Regression predictions for the test data')
```



2. Decision Tree Regression

In [21]:
```
max_depth = [5,10,15,20,25,30,35,40,45,50]
RMSE = []
mae = []
r2score = []
for n in max_depth:
    regressor = DecisionTreeRegressor(max_depth = n, random_state = 100)
    regressor.fit(X_train, y_train)
    y_prediction = regressor.predict(X_test)
    RMSE.append(sqrt(mean_squared_error(y_true = y_test, y_pred = y_predictic
    mae.append(mean_absolute_error(y_test, y_prediction))
    r2score.append(r2_score(y_test, y_prediction))

DTRegressor_results = pd.DataFrame({'max_depth':max_depth,'RMSE':RMSE, 'MAE':

print(DTRegressor_results.round(2))

fig, ax1 = plt.subplots()
ax1.plot(DTRegressor_results['max_depth'], DTRegressor_results['r2score'], 'b
ax1.set_xlabel('max_depth')
ax1.set_ylabel('r2score')
ax1.legend(['r2score'], loc ="upper right")
ax2 = ax1.twinx()
ax2.plot(DTRegressor_results['max_depth'], DTRegressor_results['MAE'], 'r--')
ax2.set_ylabel('MAE')
ax2.legend(['MAE'],loc ="upper center")
plt.show()
```

	max_depth	RMSE	MAE	r2score
0	5	197738.11	113512.93	0.71
1	10	165407.15	88514.64	0.80
2	15	178910.89	92964.29	0.76
3	20	189354.61	99370.01	0.73
4	25	188644.19	101403.71	0.74
5	30	178732.86	100080.28	0.76
6	35	182904.86	100619.37	0.75
7	40	185296.70	100791.27	0.75
8	45	185296.70	100791.27	0.75
9	50	185296.70	100791.27	0.75

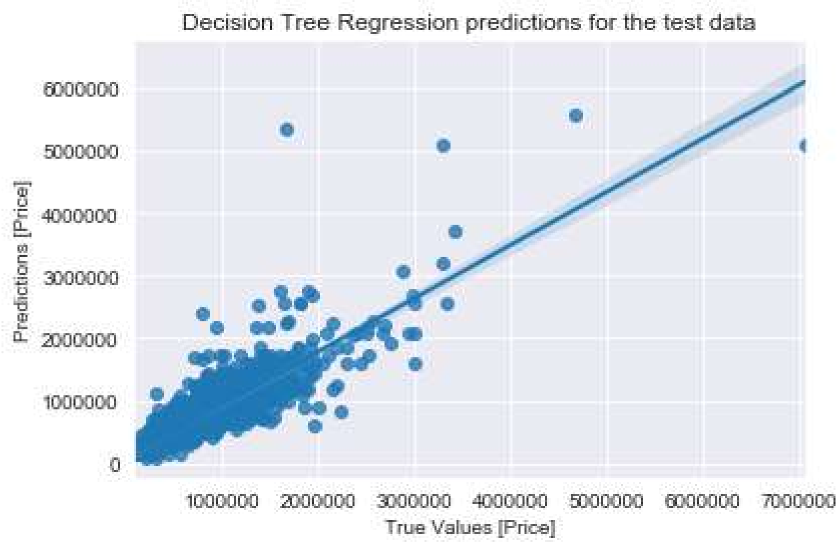


The best fitting model in this case has an r2score of 0.80 and MAE of 88514.64 with max_depth = 10.

True Value vs. Predicted value for the best fitting Decision Tree Regression model

In [22]:
```
sns.regplot(x=y_test, y=  DecisionTreeRegressor(max_depth = 10, random_state
plt.xlabel('True Values [Price]')
plt.ylabel('Predictions [Price]')
plt.title('Decision Tree Regression predictions for the test data')
```

Out[22]:
```
Text(0.5, 1.0, 'Decision Tree Regression predictions for the test data')
```

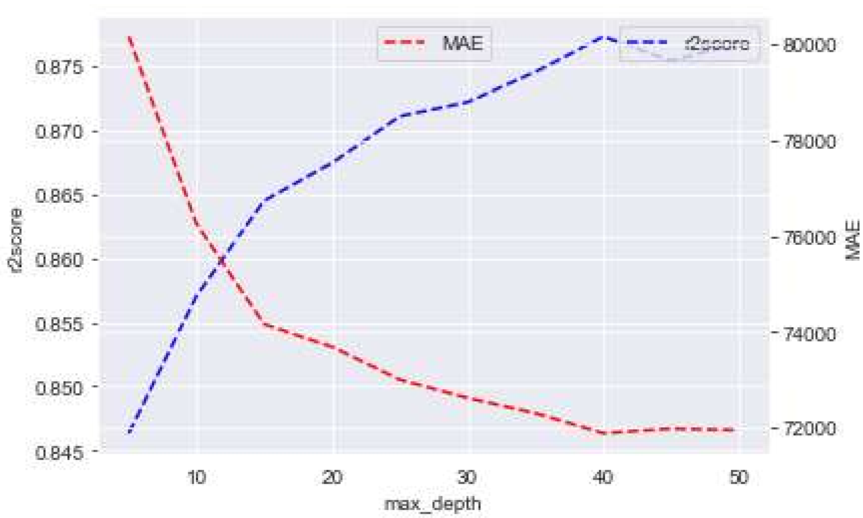


3. Random Forest Regression

In [23]:

```
n_estimators = [5,10,15,20,25,30, 35, 40, 45, 50]
RMSE = []
mae = []
r2score = []
for n in n_estimators:
    regressor = RandomForestRegressor(n_estimators = n, random_state = 100)
    regressor.fit(X_train, y_train)
    y_prediction = regressor.predict(X_test)
    RMSE.append(sqrt(mean_squared_error(y_true = y_test, y_pred = y_predictic
    mae.append(mean_absolute_error(y_test, y_prediction))
    r2score.append(r2_score(y_test, y_prediction))

RFRegression_results = pd.DataFrame({'n_estimators':n_estimators,'RMSE':RMSE,

print(RFRegression_results.round(3))

fig, ax1 = plt.subplots()
ax1.plot(RFRegression_results['n_estimators'], RFRegression_results['r2score'
ax1.set_xlabel('max_depth')
ax1.set_ylabel('r2score')
ax1.legend(['r2score'], loc ="upper right")
ax2 = ax1.twinx()
ax2.plot(RFRegression_results['n_estimators'], RFRegression_results['MAE'], '
ax2.set_ylabel('MAE')
ax2.legend(['MAE'],loc ="upper center")
plt.show()
```

```
   n_estimators        RMSE        MAE  r2score
0             5  143920.145  80161.515    0.846
1            10  138789.519  76253.773    0.857
2            15  135135.123  74162.498    0.864
3            20  133648.282  73687.905    0.867
4            25  131788.472  72999.800    0.871
5            30  131226.704  72619.218    0.872
6            35  130019.491  72299.711    0.875
7            40  128573.022  71879.346    0.877
8            45  129563.984  71974.844    0.875
9            50  128841.409  71944.298    0.877
```

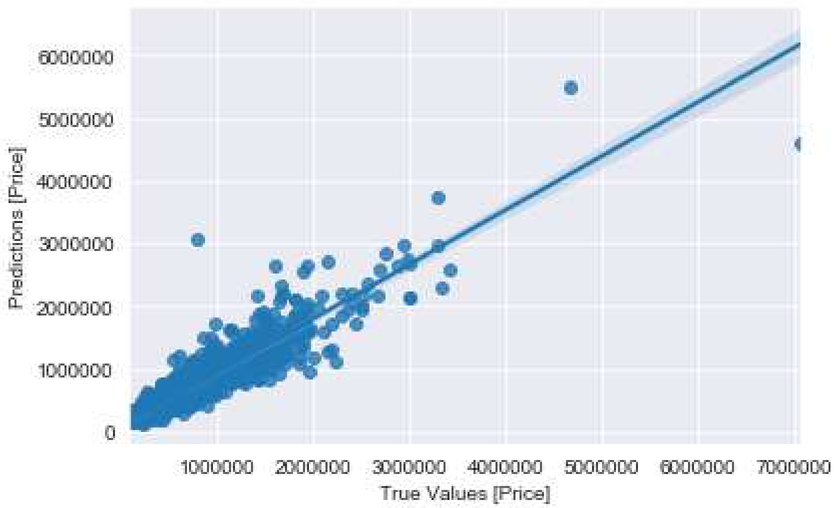


The best fitting model in this case has an r2score of 0.877 and MAE of 71879.346 with n_estimators = 40.

True Value vs. Predicted value for the best fitting Random Forest Regression model

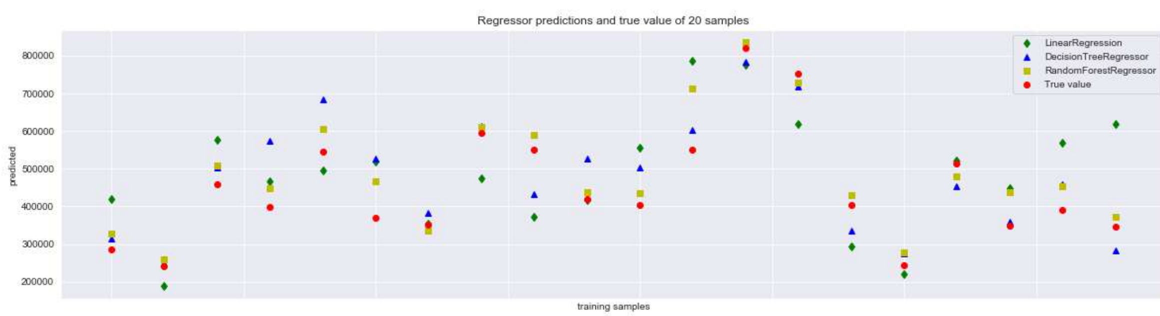
In [24]:
```
sns.regplot(x=y_test, y= RandomForestRegressor(n_estimators = 50, random_stat
plt.xlabel('True Values [Price]')
plt.ylabel('Predictions [Price]')
```

Out[24]: `Text(0, 0.5, 'Predictions [Price]')`



Comparing the different regressor models

In [25]:
```
reg1 = LinearRegression()
reg2 = DecisionTreeRegressor(max_depth = 10,  random_state = 100)
reg3 = RandomForestRegressor(n_estimators = 40, random_state = 100)

reg1.fit(X_train, y_train)
reg2.fit(X_train, y_train)
reg3.fit(X_train, y_train)

pred1 = reg1.predict(X_test[:20])
pred2 = reg2.predict(X_test[:20])
pred3 = reg3.predict(X_test[:20])


plt.figure(figsize=(20,5))
plt.plot(pred1, 'gd', label='LinearRegression')
plt.plot(pred2, 'b^', label='DecisionTreeRegressor')
plt.plot(pred3, 'ys', label='RandomForestRegressor')
plt.plot(y_test[:20], 'ro', label = 'True value')

plt.tick_params(axis='x', which='both', bottom=False, top=False,
                labelbottom=False)
plt.ylabel('predicted')
plt.xlabel('training samples')
plt.legend(loc="best")
plt.title('Regressor predictions and true value of 20 samples')

plt.show()
```



The above graph shows the house price predictions with the different regressor models used and the actual price for the first 20 samples in the test dataset.

The highest r2score (0.877) is obtained with Random Forest Regression model.