HOUSING PRICE PREDICTION PROJECT

By

Alivia Dasgupta

Acknowledgement

Firstly, I would like to thank FlipRobo Technologies for giving me the opportunity to work on this project. This project includes mentioning of all the references, research papers, data sources, professionals and other resources that helped me and guided me in completion project.

INTRODUCTION

Business Problem Framing

The main objective of this project is to model the price of houses with the available independent variables. This model will then be used by the management to understand how exactly the prices vary with the variables. They can accordingly manipulate the strategy of the firm and concentrate on areas that will yield high returns. Further, the model will be a good way for the management to understand the pricing dynamics of a new market.

Conceptual Background of the Domain Problem

Houses are one of the necessary need of each and every person around the globe and therefore housing and real estate market is one of the markets which is one of the major contributors in the world's economy. It is a very large market and there are various companies working in the domain. Data science comes as a very important tool to solve problems in the domain to help the companies increase their overall revenue, profits, improving their marketing strategies and focusing on changing trends in house sales and purchases. Predictive modelling, Market mix modelling, recommendation systems are some of the machine learning techniques used for achieving the business goals for housing companies. Our problem is related to one such housing company.

Technical Requirements

➢ Data contains 1460 entries each having 81 variables.

➢ Data contains Null values. We need to treat them using the domain knowledge and your own understanding.

➢ Extensive EDA has to be performed to gain relationships of variable and price.

➢ Data contains numerical as well as categorical variable. We need to handle them accordingly.

➢ We have to build Machine Learning methods, apply regularization and determine the optimal values for HyperParameters.

➢ We need to find important features which affect the price positively or negatively.

➢ Two datasets are being provided to us (test.csv, train.csv).

ADVANTAGES:

1. The objective behind to take this project is to implement the required data science skills.

2. Improve the analytical thinking.

3. Get into the real world problem solving mechanics.

Analytical Problem Framing

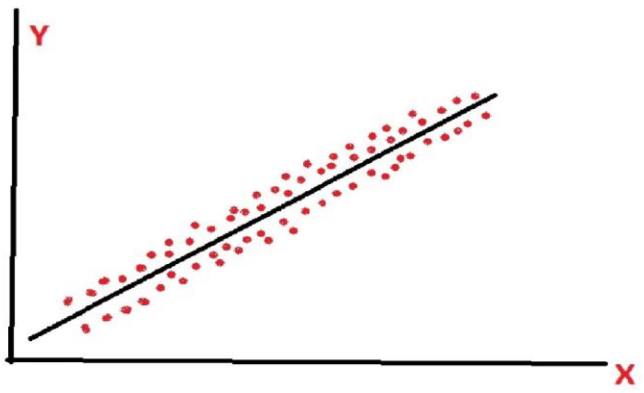
Mathematical Modeling of the Problem:

This is a Regression problem, where our end goal is to predict the Prices of House, based on given data provided in the dataset. We have divided the provided dataset into Training a nd Testing phases. A Regression Model will be built and trained using the Training data and the Test data is used to predict the outcomes. This will be compared with available test results to find how our model has performed.

We are using Mean Absolute Error, Root Mean Square Error, and ‘R2_Score’ to determine the best model among, ❖ Linear Regression ❖ Decision Tree Regression ❖ Random Forest Regression ❖ K Neighbors Regression ❖ Lasso Regression The best results were obtained using Lasso Regression. So, let’s discuss a little bit about it.

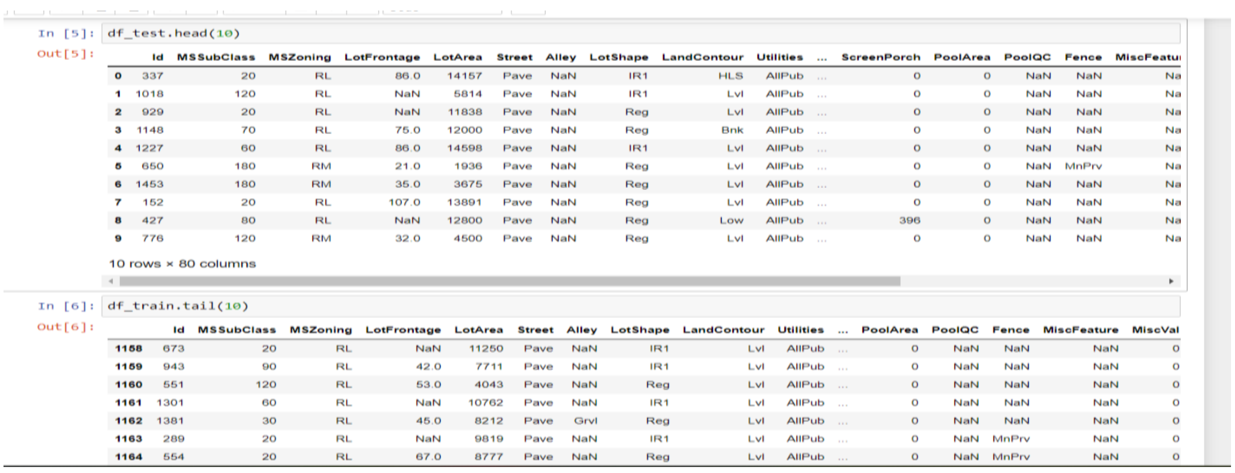
In a simple regression problem (a single xand a single y), the form of the model would be:

y = B0 + B1*x where B0 —intercept, B1 —coefficient, x — independent variable y — output or the dependent variable. In higher dimensions when we have more than one input (x), The General equation for a Multiple linear regression with p — independent variables: Y=B0 + B1 * X1 + B2 * X2 +............+ Bp * Xp + E(Random Error or Noise).



Data Sources and their formats

A US-based housing company named Surprise Housing has decided to enter the Australian market. The company uses data analytic to purchase houses at a price below their actual values and flip them at a higher price. For the same purpose, the company has collected a data set from the sale of houses in Australia. The data is provided in the CSV file below. The dataset contains 1460 rows and 81 columns (including the train dataset and test dataset). The top 5 rows of the dataset are:

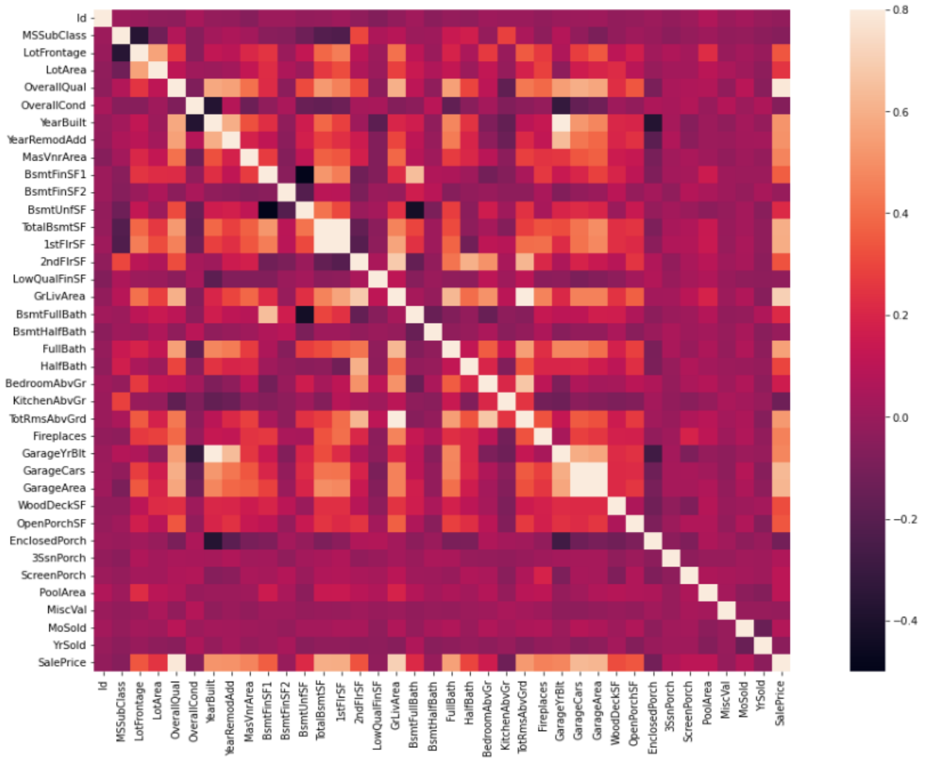


```
In [5]: df_test.head(10)
Out[5]:
     Id  MSSubClass  MSZoning  LotFrontage  LotArea  Street  Alley  LotShape  LandContour  Utilities  ...  ScreenPorch  PoolArea  PoolQC  Fence  MiscFeatu
0   337          20        RL         86.0    14157    Pave    NaN       IR1          HLS     AllPub  ...            0         0     NaN    NaN         Na
1  1018         120        RL          NaN     5814    Pave    NaN       IR1          Lvl     AllPub  ...            0         0     NaN    NaN         Na
2   929          20        RL          NaN    11838    Pave    NaN       Reg          Lvl     AllPub  ...            0         0     NaN    NaN         Na
3  1148          70        RL         75.0    12000    Pave    NaN       Reg          Bnk     AllPub  ...            0         0     NaN    NaN         Na
4  1227          60        RL         86.0    14598    Pave    NaN       IR1          Lvl     AllPub  ...            0         0     NaN    NaN         Na
5   650         180        RM         21.0     1936    Pave    NaN       Reg          Lvl     AllPub  ...            0         0     NaN  MnPrv         Na
6  1453         180        RM         35.0     3675    Pave    NaN       Reg          Lvl     AllPub  ...            0         0     NaN    NaN         Na
7   152          20        RL        107.0    13891    Pave    NaN       Reg          Lvl     AllPub  ...            0         0     NaN    NaN         Na
8   427          80        RL          NaN    12800    Pave    NaN       Reg          Low     AllPub  ...          396         0     NaN    NaN         Na
9   776         120        RM         32.0     4500    Pave    NaN       Reg          Lvl     AllPub  ...            0         0     NaN    NaN         Na

10 rows × 80 columns

In [6]: df_train.tail(10)
Out[6]:
        Id  MSSubClass  MSZoning  LotFrontage  LotArea  Street  Alley  LotShape  LandContour  Utilities  ...  PoolArea  PoolQC  Fence  MiscFeature  MiscVal
1158   673          20        RL          NaN    11250    Pave    NaN       IR1          Lvl     AllPub  ...         0     NaN    NaN          NaN        0
1159   943          90        RL         42.0     7711    Pave    NaN       IR1          Lvl     AllPub  ...         0     NaN    NaN          NaN        0
1160   551         120        RL         53.0     4043    Pave    NaN       Reg          Lvl     AllPub  ...         0     NaN    NaN          NaN        0
1161  1301          60        RL          NaN    10762    Pave    NaN       IR1          Lvl     AllPub  ...         0     NaN    NaN          NaN        0
1162  1381          30        RL         45.0     8212    Pave   Grvl       Reg          Lvl     AllPub  ...         0     NaN    NaN          NaN        0
1163   289          20        RL          NaN     9819    Pave    NaN       IR1          Lvl     AllPub  ...         0     NaN  MnPrv          NaN        0
1164   554          20        RL         67.0     8777    Pave    NaN       Reg          Lvl     AllPub  ...         0     NaN  MnPrv          NaN        0
```

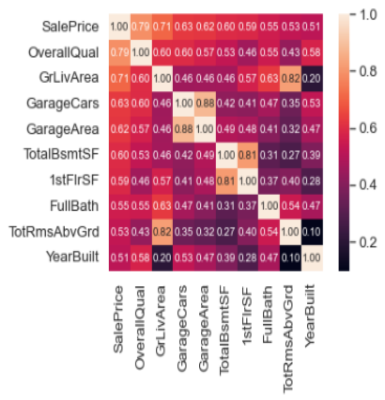
Data Preprocessing Done

As our dataset contains null values (missing values) so we have replace the missing values with the required values. Details are mentioned below

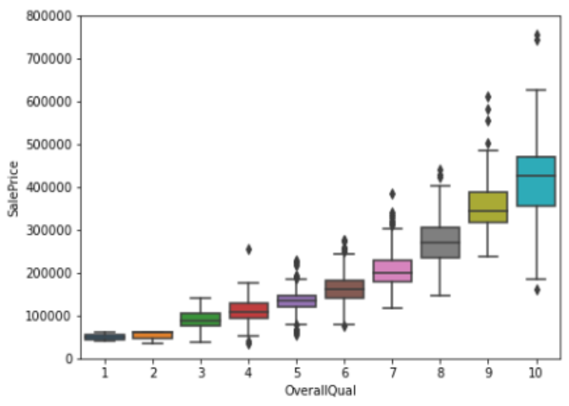
```
In [43]: corrmat = df_train.corr()
         f, ax = plt.subplots(figsize=(20, 12))
         sns.heatmap(corrmat, vmax=.8, square=True);
```



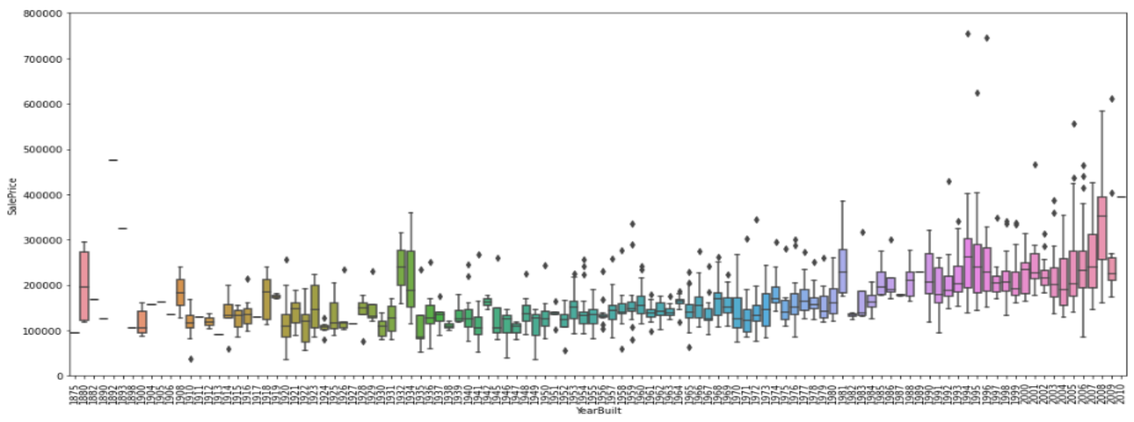
```
In [46]: k = 10 #number of variables for heatmap
         cols = corrmat.nlargest(k, 'SalePrice')['SalePrice'].index
         cm = np.corrcoef(df_train[cols].values.T)
         sns.set(font_scale=1.25)
         hm = sns.heatmap(cm, cbar=True, annot=True, square=True, fmt='.2f', annot_kws={'size': 10}, yticklabels=cols.values, xticklabels=
         plt.show()
```



```
In [39]: var = 'OverallQual'
         data = pd.concat([df_train['SalePrice'], df_train[var]], axis=1)
         f, ax = plt.subplots(figsize=(7, 5))
         fig = sns.boxplot(x=var, y="SalePrice", data=data)
         fig.axis(ymin=0, ymax=800000);
```



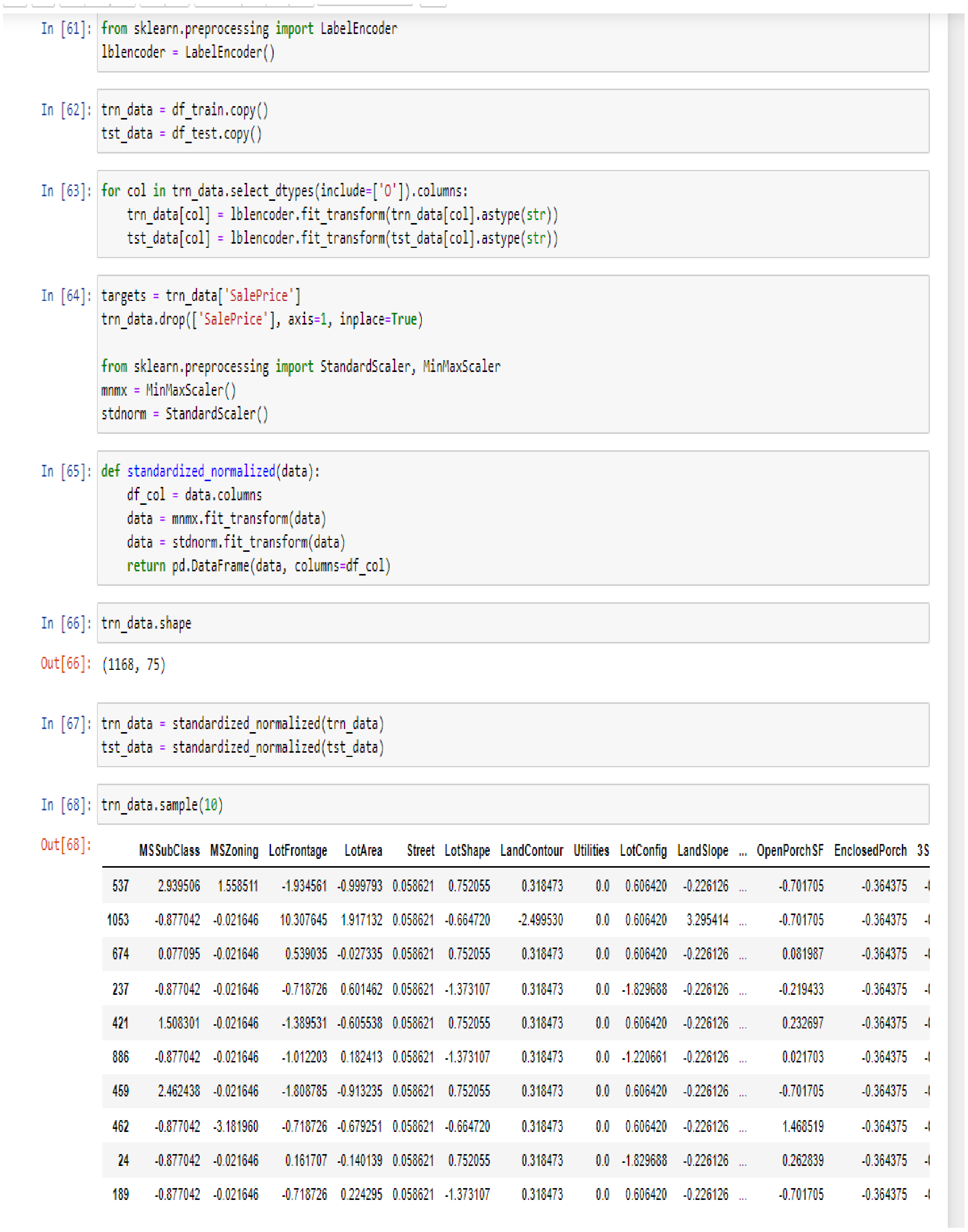
```
In [38]: var = 'YearBuilt'
         data = pd.concat([df_train['SalePrice'], df_train[var]], axis=1)
         f, ax = plt.subplots(figsize=(18, 8))
         fig = sns.boxplot(x=var, y="SalePrice", data=data)
         fig.axis(ymin=0, ymax=800000);
         plt.xticks(rotation=90);
```



Summary of EDA

1. GrLivArea' and 'TotalBsmtSF' seem to be linearly related with 'SalePrice'. Both relationships are positive, which means that as one variable increases, the other also increases. In the case of 'TotalBsmtSF', we can see that the slope of the linear relationship is particularly high.
2. 'OverallQual' and 'YearBuilt' also seem to be related with 'SalePrice'. The relationship seems to be stronger in the case of 'OverallQual', where the box plot shows how sales prices increase with the overall quality. We just analysed four variables, but there are many other that we should analyse. The trick here seems to be the choice of the right features (feature selection) and not the definition of complex relationships between them (feature engineering). That said, let's separate the wheat from the chaff

Models



```
In [61]: from sklearn.preprocessing import LabelEncoder
         lblencoder = LabelEncoder()
```

```
In [62]: trn_data = df_train.copy()
         tst_data = df_test.copy()
```

```
In [63]: for col in trn_data.select_dtypes(include=['O']).columns:
             trn_data[col] = lblencoder.fit_transform(trn_data[col].astype(str))
             tst_data[col] = lblencoder.fit_transform(tst_data[col].astype(str))
```

```
In [64]: targets = trn_data['SalePrice']
         trn_data.drop(['SalePrice'], axis=1, inplace=True)

         from sklearn.preprocessing import StandardScaler, MinMaxScaler
         mnmx = MinMaxScaler()
         stdnorm = StandardScaler()
```

```
In [65]: def standardized_normalized(data):
             df_col = data.columns
             data = mnmx.fit_transform(data)
             data = stdnorm.fit_transform(data)
             return pd.DataFrame(data, columns=df_col)
```

```
In [66]: trn_data.shape
```

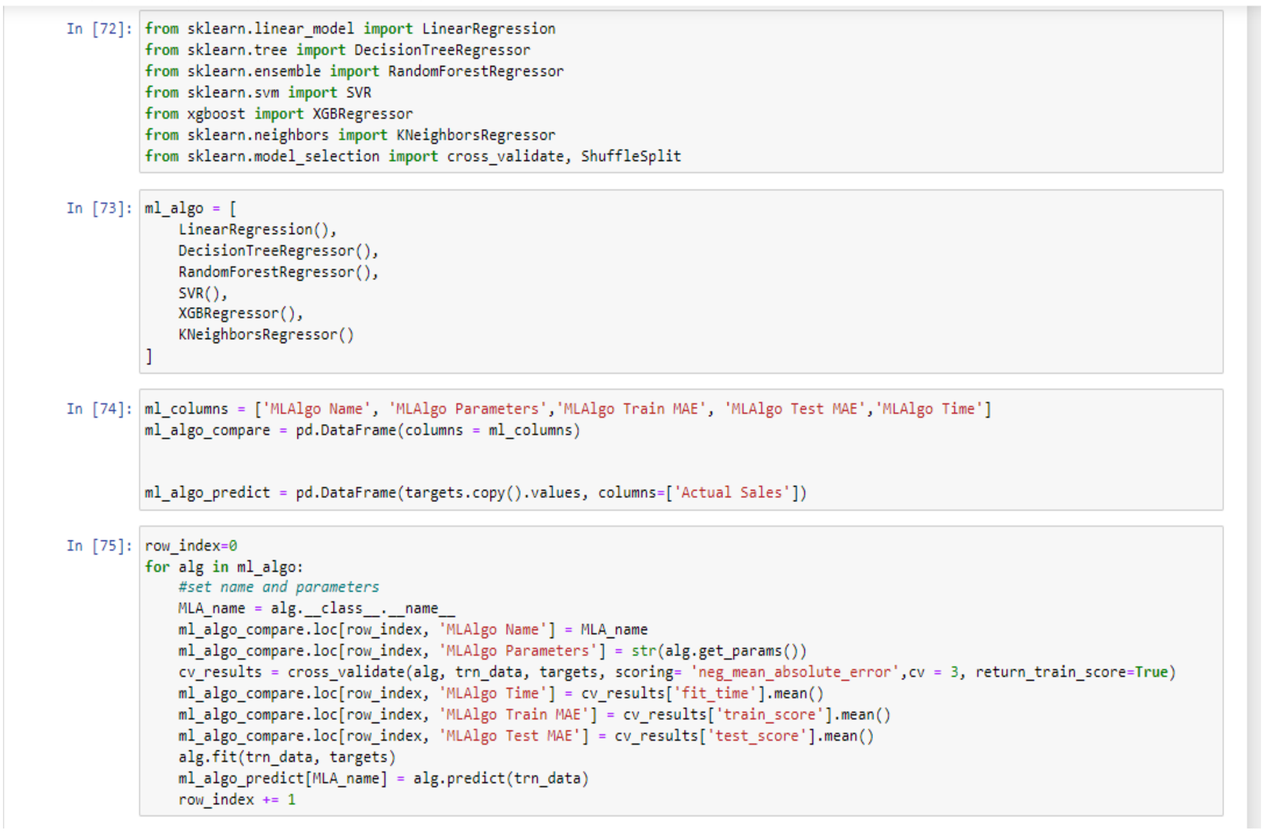
```
Out[66]: (1168, 75)
```

```
In [67]: trn_data = standardized_normalized(trn_data)
         tst_data = standardized_normalized(tst_data)
```

```
In [68]: trn_data.sample(10)
```

Out[68]:

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	Utilities	LotConfig	LandSlope	...	OpenPorchSF	EnclosedPorch	3S
537	2.939506	1.558511	-1.934561	-0.999793	0.058621	0.752055	0.318473	0.0	0.606420	-0.226126	...	-0.701705	-0.364375	-
1053	-0.877042	-0.021646	10.307645	1.917132	0.058621	-0.664720	-2.499530	0.0	0.606420	3.295414	...	-0.701705	-0.364375	-
674	0.077095	-0.021646	0.539035	-0.027335	0.058621	0.752055	0.318473	0.0	0.606420	-0.226126	...	0.081987	-0.364375	-
237	-0.877042	-0.021646	-0.718726	0.601462	0.058621	-1.373107	0.318473	0.0	-1.829688	-0.226126	...	-0.219433	-0.364375	-
421	1.508301	-0.021646	-1.389531	-0.605538	0.058621	0.752055	0.318473	0.0	0.606420	-0.226126	...	0.232697	-0.364375	-
886	-0.877042	-0.021646	-1.012203	0.182413	0.058621	-1.373107	0.318473	0.0	-1.220661	-0.226126	...	0.021703	-0.364375	-
459	2.462438	-0.021646	-1.808785	-0.913235	0.058621	0.752055	0.318473	0.0	0.606420	-0.226126	...	-0.701705	-0.364375	-
462	-0.877042	-3.181960	-0.718726	-0.679251	0.058621	-0.664720	0.318473	0.0	0.606420	-0.226126	...	1.468519	-0.364375	-
24	-0.877042	-0.021646	0.161707	-0.140139	0.058621	0.752055	0.318473	0.0	-1.829688	-0.226126	...	0.262839	-0.364375	-
189	-0.877042	-0.021646	-0.718726	0.224295	0.058621	-1.373107	0.318473	0.0	0.606420	-0.226126	...	-0.701705	-0.364375	-

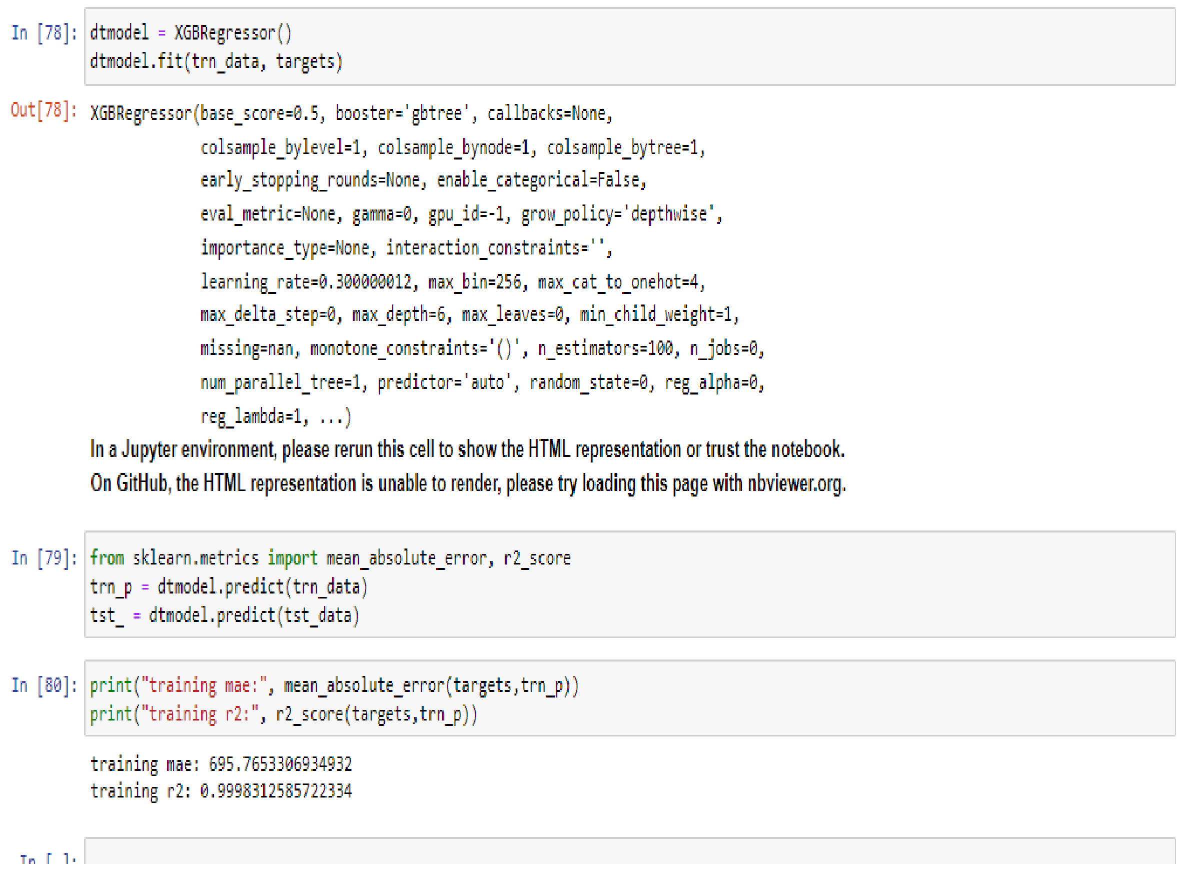


```
In [72]: from sklearn.linear_model import LinearRegression
         from sklearn.tree import DecisionTreeRegressor
         from sklearn.ensemble import RandomForestRegressor
         from sklearn.svm import SVR
         from xgboost import XGBRegressor
         from sklearn.neighbors import KNeighborsRegressor
         from sklearn.model_selection import cross_validate, ShuffleSplit
```

```
In [73]: ml_algo = [
             LinearRegression(),
             DecisionTreeRegressor(),
             RandomForestRegressor(),
             SVR(),
             XGBRegressor(),
             KNeighborsRegressor()
         ]
```

```
In [74]: ml_columns = ['MLAlgo Name', 'MLAlgo Parameters','MLAlgo Train MAE', 'MLAlgo Test MAE','MLAlgo Time']
         ml_algo_compare = pd.DataFrame(columns = ml_columns)


         ml_algo_predict = pd.DataFrame(targets.copy().values, columns=['Actual Sales'])
```

```
In [75]: row_index=0
         for alg in ml_algo:
             #set name and parameters
             MLA_name = alg.__class__.__name__
             ml_algo_compare.loc[row_index, 'MLAlgo Name'] = MLA_name
             ml_algo_compare.loc[row_index, 'MLAlgo Parameters'] = str(alg.get_params())
             cv_results = cross_validate(alg, trn_data, targets, scoring= 'neg_mean_absolute_error',cv = 3, return_train_score=True)
             ml_algo_compare.loc[row_index, 'MLAlgo Time'] = cv_results['fit_time'].mean()
             ml_algo_compare.loc[row_index, 'MLAlgo Train MAE'] = cv_results['train_score'].mean()
             ml_algo_compare.loc[row_index, 'MLAlgo Test MAE'] = cv_results['test_score'].mean()
             alg.fit(trn_data, targets)
             ml_algo_predict[MLA_name] = alg.predict(trn_data)
             row_index += 1
```



```
In [78]: dtmodel = XGBRegressor()
         dtmodel.fit(trn_data, targets)
```

```
Out[78]: XGBRegressor(base_score=0.5, booster='gbtree', callbacks=None,
                      colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
                      early_stopping_rounds=None, enable_categorical=False,
                      eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
                      importance_type=None, interaction_constraints='',
                      learning_rate=0.300000012, max_bin=256, max_cat_to_onehot=4,
                      max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight=1,
                      missing=nan, monotone_constraints='()', n_estimators=100, n_jobs=0,
                      num_parallel_tree=1, predictor='auto', random_state=0, reg_alpha=0,
                      reg_lambda=1, ...)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
In [79]: from sklearn.metrics import mean_absolute_error, r2_score
         trn_p = dtmodel.predict(trn_data)
         tst_ = dtmodel.predict(tst_data)
```

```
In [80]: print("training mae:", mean_absolute_error(targets,trn_p))
         print("training r2:", r2_score(targets,trn_p))

         training mae: 695.7653306934932
         training r2: 0.9998312585722334
```

```
In [ ]:
```

Conclusion

We have observed about the variables, we analysed 'SalePrice' alone and with the most correlated variables, we dealt with missing data and outliers, we tested some of the fundamental statistical assumptions and we even transformed categorial variables into dummy variables. That's a lot of work that Python helped us make easier.