Udacity AWS Machine Learning Nanodegree

Capstone Project

Predicting House Prices with Machine Learning Algorithms

Pakiza Valizada

March 2023

Project Overview and Problem Statement

Real estate market is considered one of the most volatile industries since house prices are very prone to change depending on economic conditions of the countries. According to Maslow's "Hierarchy of Needs", house is considered one of the most essential needs of humans. Therefore, the main aim of this project is to predict the house prices based on various features (area of the house, number of the rooms, number of the bathrooms, location, availability and size of garage and etc.) The AWS SageMaker Studio (Python 3 - jupyter notebook) and "ml.m5.xlarge" instance type are used in order to run the model.

The paper also aims to determine which variables are correlated and how these variables can be used to predict the house prices. Client house sellers want to buy a house at a reasonable price with the highest return, at the same time, house buyers want to make sure that they want to get a fair price on houses.

I'll use various regression techniques to determine the prices of houses based on their features.

Algorithms and techniques

Random Forest, Gradient Boosting Machine, Lasso, Elastic Net are used as a machine learning algorithms. Feature engineering and data preprocessing are utilized to improve the performance of the algorithm. Combined dataset was divided into train (70%) and validation (30%) set before building the model.

Metrics

Mean Absolute Error (MAE) is used as a model evaluation metric for this project. Basically, the "MAE" stands for the average of each prediction error's absolute values over all instances of the test data set.

Analysis

Data Exploration

Dataset and Inputs can be found on Kaggle's website:

https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques/data

Dataset consists of the files below: [1]

- **train.csv** - the training set
- **test.csv** - the test set
- **data_description.txt** - full description of each column, originally prepared by Dean De Cock but lightly edited to match the column names used here
- **sample_submission.csv** - a benchmark submission from a linear regression on year and month of sale, lot square footage, and number of bedrooms

Data description file consist of data fields below:

- **SalePrice** - the property's sale price in dollars. This is the target variable that we're trying to predict.
- **MSSubClass**: The building class
- **MSZoning**: The general zoning classification
- **LotFrontage**: Linear feet of street connected to property
- **LotArea**: Lot size in square feet
- **Street**: Type of road access
- **Alley**: Type of alley access
- **LotShape**: General shape of property
- **LandContour**: Flatness of the property
- **Utilities**: Type of utilities available
- **LotConfig**: Lot configuration
- **LandSlope**: Slope of property
- **Neighborhood**: Physical locations within Ames city limits
- **Condition1**: Proximity to main road or railroad
- **Condition2**: Proximity to main road or railroad (if a second is present)
- **BldgType**: Type of dwelling
- **HouseStyle**: Style of dwelling
- **OverallQual**: Overall material and finish quality

- **OverallCond**: Overall condition rating
- **YearBuilt**: Original construction date
- **YearRemodAdd**: Remodel date
- **RoofStyle**: Type of roof
- **RoofMatl**: Roof material
- **Exterior1st**: Exterior covering on house
- **Exterior2nd**: Exterior covering on house (if more than one material)
- **MasVnrType**: Masonry veneer type
- **MasVnrArea**: Masonry veneer area in square feet
- **ExterQual**: Exterior material quality
- **ExterCond**: Present condition of the material on the exterior
- **Foundation**: Type of foundation
- **BsmtQual**: Height of the basement
- **BsmtCond**: General condition of the basement
- **BsmtExposure**: Walkout or garden level basement walls
- **BsmtFinType1**: Quality of basement finished area
- **BsmtFinSF1**: Type 1 finished square feet
- **BsmtFinType2**: Quality of second finished area (if present)
- **BsmtFinSF2**: Type 2 finished square feet
- **BsmtUnfSF**: Unfinished square feet of basement area
- **TotalBsmtSF**: Total square feet of basement area
- **Heating**: Type of heating
- **HeatingQC**: Heating quality and condition
- **CentralAir**: Central air conditioning
- **Electrical**: Electrical system
- **1stFlrSF**: First Floor square feet
- **2ndFlrSF**: Second floor square feet
- **LowQualFinSF**: Low quality finished square feet (all floors)

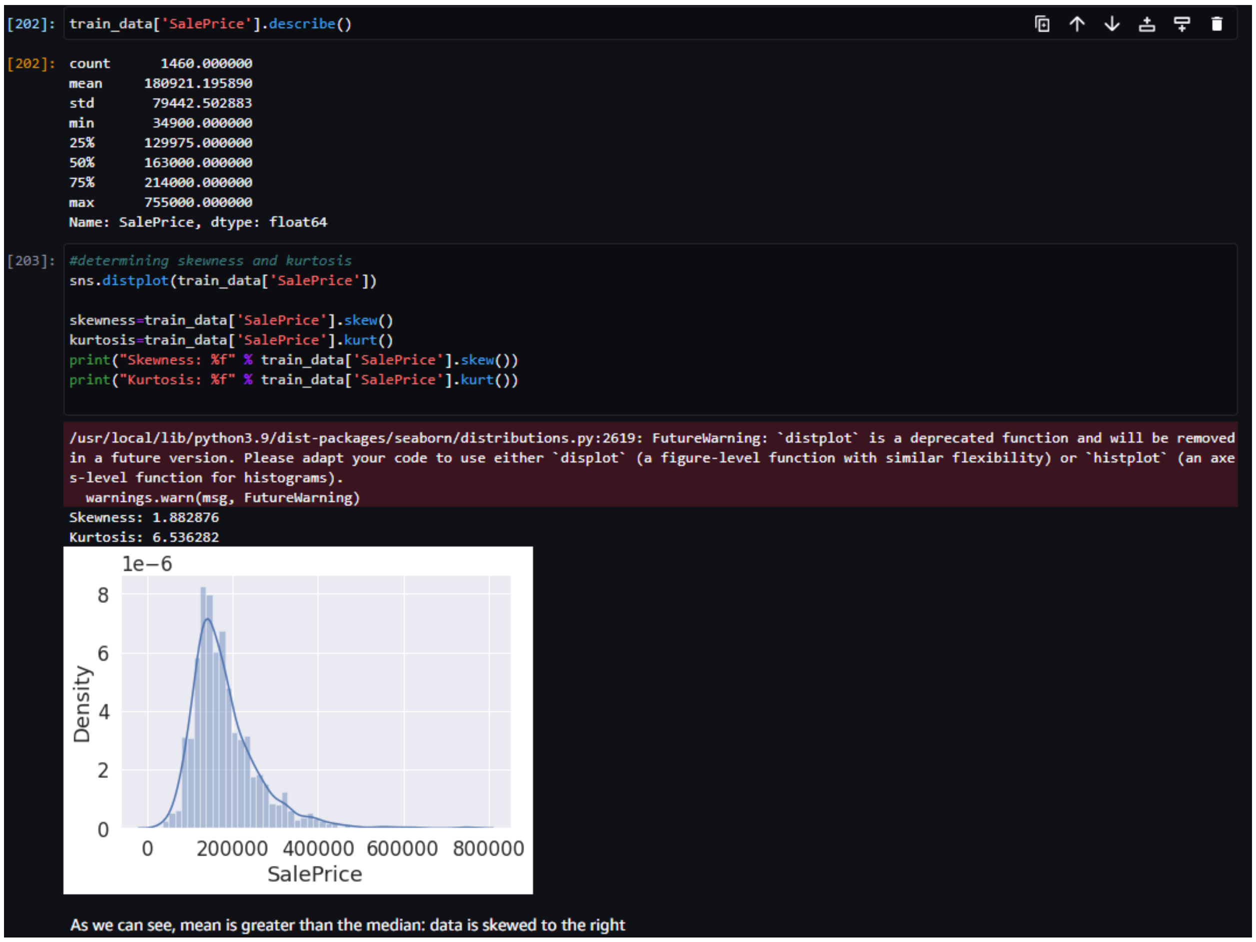
- **GrLivArea**: Above grade (ground) living area square feet
- **BsmtFullBath**: Basement full bathrooms
- **BsmtHalfBath**: Basement half bathrooms
- **FullBath**: Full bathrooms above grade
- **HalfBath**: Half baths above grade
- **Bedroom**: Number of bedrooms above basement level
- **Kitchen**: Number of kitchens
- **KitchenQual**: Kitchen quality
- **TotRmsAbvGrd**: Total rooms above grade (does not include bathrooms)
- **Functional**: Home functionality rating
- **Fireplaces**: Number of fireplaces
- **FireplaceQu**: Fireplace quality
- **GarageType**: Garage location
- **GarageYrBlt**: Year garage was built
- **GarageFinish**: Interior finish of the garage
- **GarageCars**: Size of garage in car capacity
- **GarageArea**: Size of garage in square feet
- **GarageQual**: Garage quality
- **GarageCond**: Garage condition
- **PavedDrive**: Paved driveway
- **WoodDeckSF**: Wood deck area in square feet
- **OpenPorchSF**: Open porch area in square feet
- **EnclosedPorch**: Enclosed porch area in square feet
- **3SsnPorch**: Three season porch area in square feet
- **ScreenPorch**: Screen porch area in square feet
- **PoolArea**: Pool area in square feet
- **PoolQC**: Pool quality
- **Fence**: Fence quality

- **MiscFeature**: Miscellaneous feature not covered in other categories
- **MiscVal**: $Value of miscellaneous feature
- **MoSold**: Month Sold
- **YrSold**: Year Sold
- **SaleType**: Type of sale
- **SaleCondition**: Condition of sale

The “SalesPrice” are used as a dependent variable in the model.   To determine the distribution of the labels in the dataset, probability plot, joint plot, boxplot, barplot, etc. are used.

Exploratory Visualization

The “describe” method is used to display certain fundamental statistical information, such as percentile, mean, standard deviation, etc., for a data frame or a collection of numerical numbers. As we can see from the distribution plot of “SalePrice” variable in the train dataset, “SalePrice” variable is skewed to the right. After cleaning the data and creating new variables, I normalized the distribution by using log transformation.

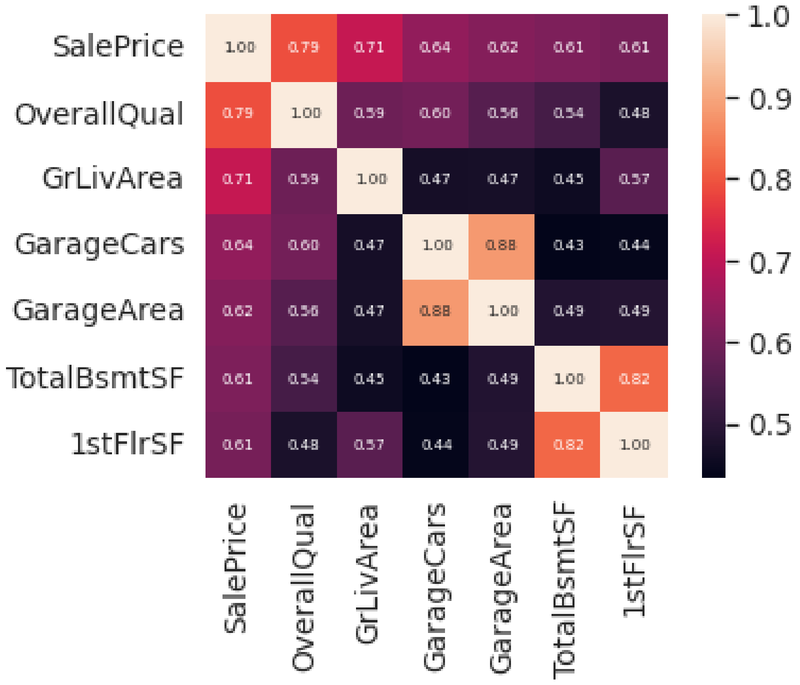


```
[202]: train_data['SalePrice'].describe()

[202]: count      1460.000000
       mean     180921.195890
       std       79442.502883
       min       34900.000000
       25%      129975.000000
       50%      163000.000000
       75%      214000.000000
       max      755000.000000
       Name: SalePrice, dtype: float64

[203]: #determining skewness and kurtosis
       sns.distplot(train_data['SalePrice'])

       skewness=train_data['SalePrice'].skew()
       kurtosis=train_data['SalePrice'].kurt()
       print("Skewness: %f" % train_data['SalePrice'].skew())
       print("Kurtosis: %f" % train_data['SalePrice'].kurt())

       /usr/local/lib/python3.9/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed
       in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axe
       s-level function for histograms).
         warnings.warn(msg, FutureWarning)
       Skewness: 1.882876
       Kurtosis: 6.536282
```

As we can see, mean is greater than the median: data is skewed to the right

To determine which variables are highly correlated with housing prices("SalePrice"), correlation matrix heatmap was used. The results of top highly correlated variables with housing pricese are depicted below:

1. OverallQual: Rates the overall material and finish of the house (1 = Very Poor, 10 = Very Excellent)
2. GrLivArea: Above grade (ground) living area square feet
3. GarageCars: Size of garage in car capacity
4. GarageArea: Size of garage in square feet



5. TotalBsmtSF: Total square feet of basement area

Benchmark

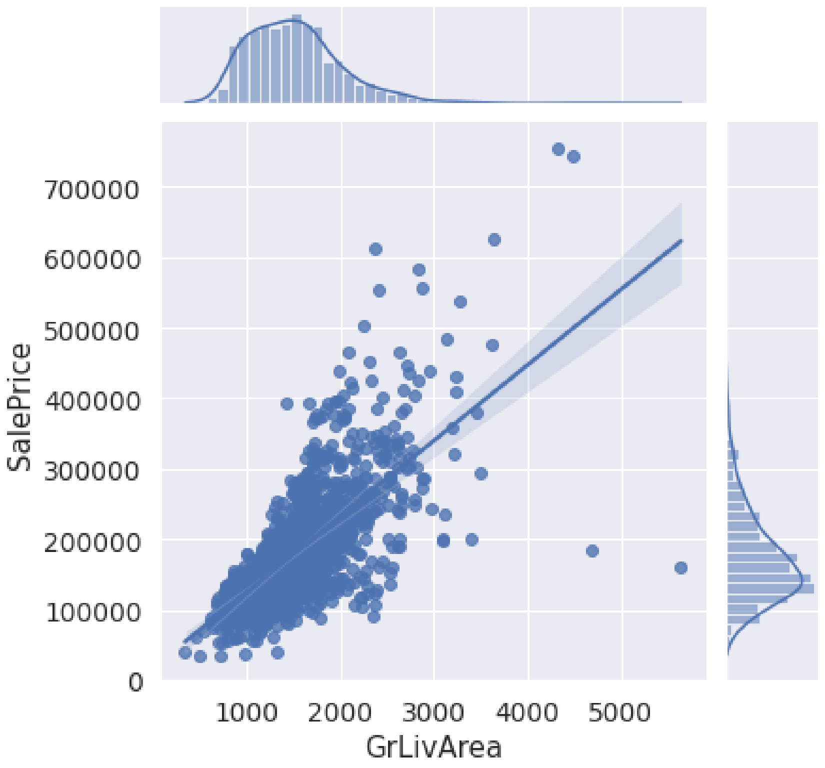
For this model, I used the algorithms outlined in “Winky K.O. Ho, Bo-Sin Tang & Siu Wai Wong (2021) Predicting property prices with machine learning algorithms, Journal of Property Research.”[2]

Methodology

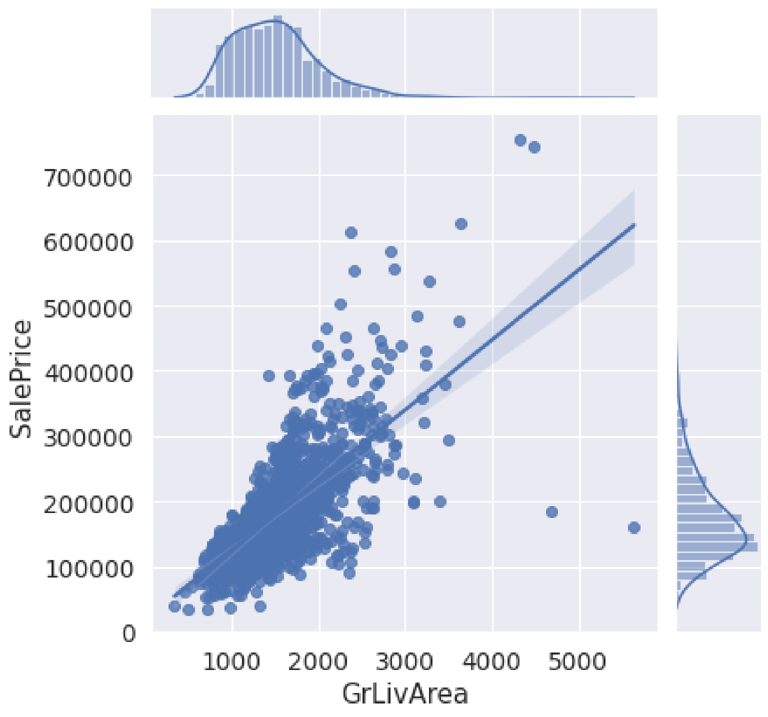
Data Preprocessing and Feature Engineering

- **Removing outliers:**

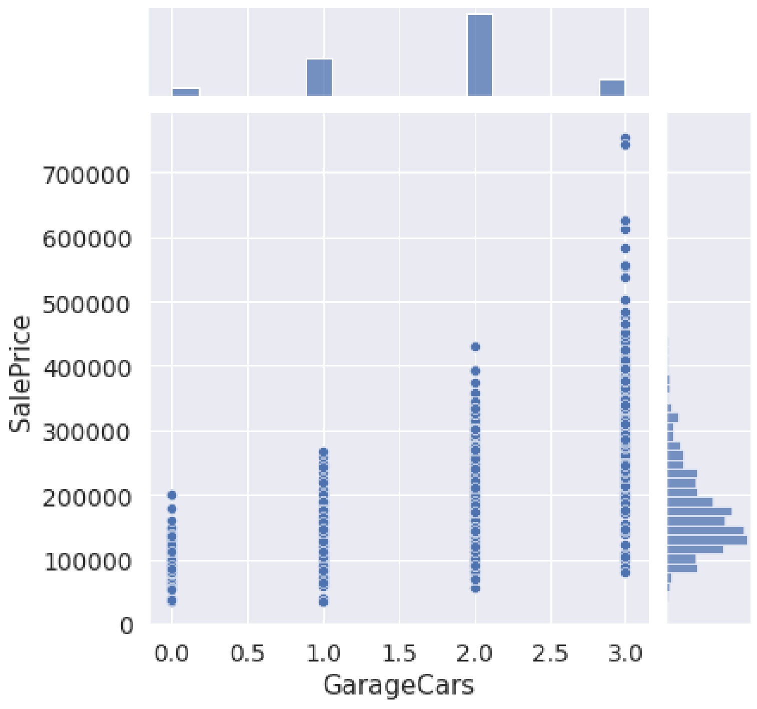
As we see from the joint plot below, there are two outlier points in the data which depict that if “Above grade (ground) living area” is more than 4000 square feet, Sale Price decreases to below 200 000$, therefore I decided to remove these outliers manually.



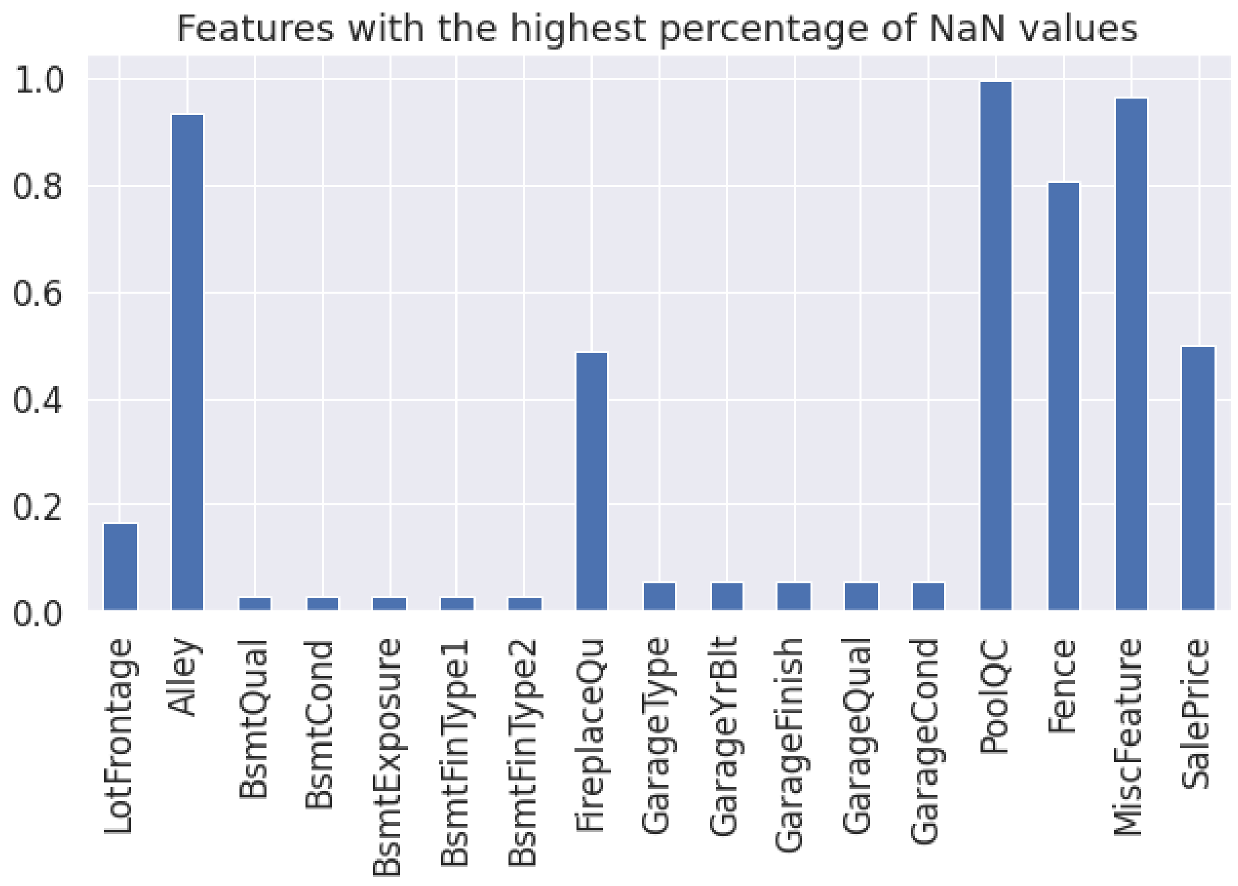
Additionally, based on the plot below, we can see that there are also two outliers in the dataset, which indicate that if the area of the garage is greater than 1000 square feet, Sale Price falls below 300 000. For that reason, I removed the outliers.

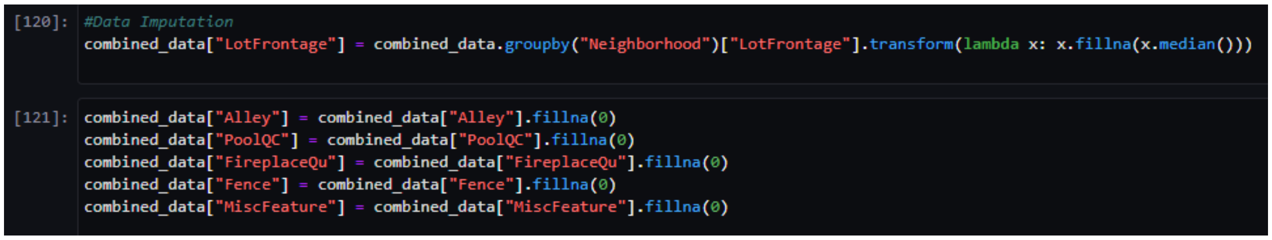


Moreover, according to the results of the plot below, we can see that 4-car garages result in less Sale Price. Therefore, I also decided to remove these outliers manually.



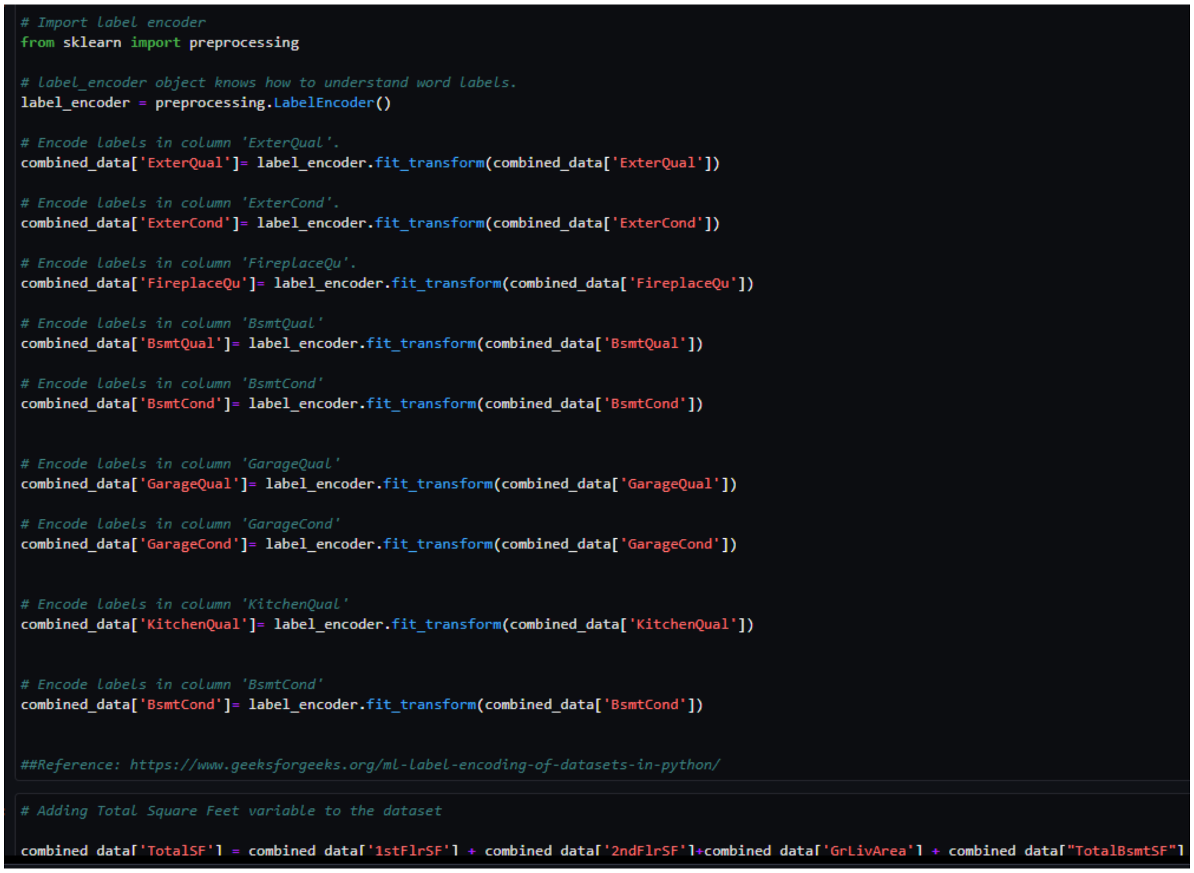
- **Data imputation**




```
[120]: #Data Imputation
       combined_data["LotFrontage"] = combined_data.groupby("Neighborhood")["LotFrontage"].transform(lambda x: x.fillna(x.median()))

[121]: combined_data["Alley"] = combined_data["Alley"].fillna(0)
       combined_data["PoolQC"] = combined_data["PoolQC"].fillna(0)
       combined_data["FireplaceQu"] = combined_data["FireplaceQu"].fillna(0)
       combined_data["Fence"] = combined_data["Fence"].fillna(0)
       combined_data["MiscFeature"] = combined_data["MiscFeature"].fillna(0)
```

As stated in "data_description.txt", "MSSubClass" variable identifies the type of dwelling involved in the sale. However, in the dataset, the variable is interpreted as a numerical value instead of the categorical value. Therefore, I converted it to the categorical variable.

- **Label Encoding**

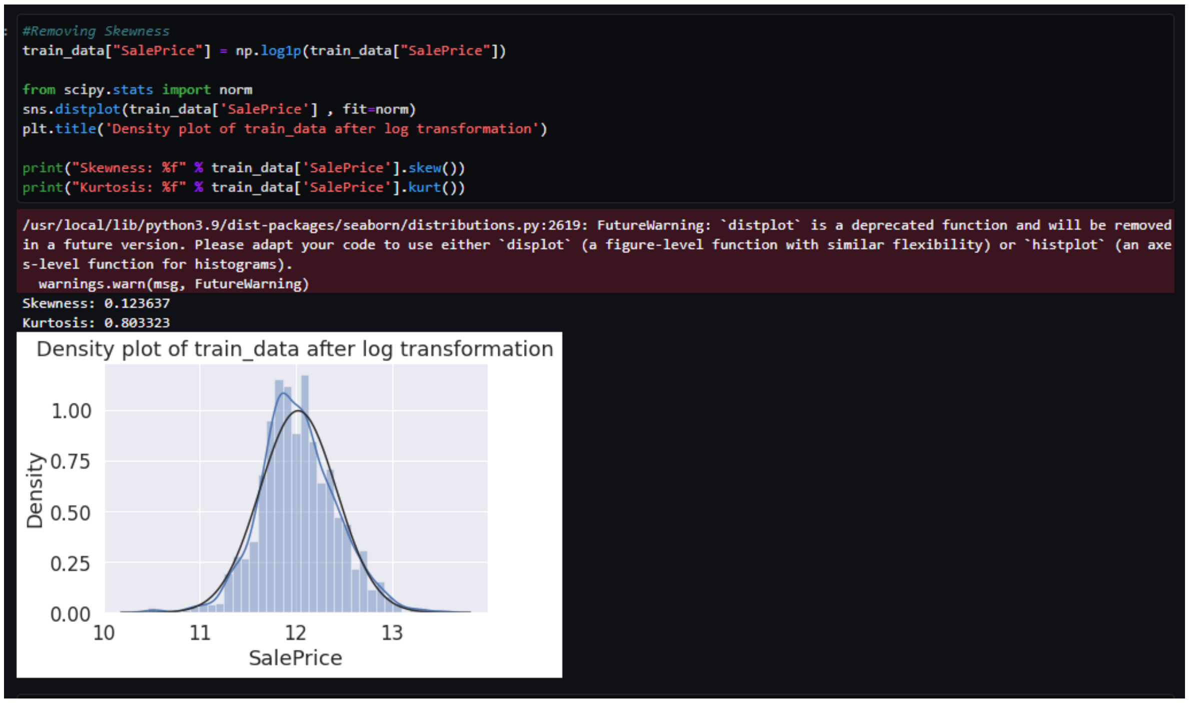


```
# Import label encoder
from sklearn import preprocessing

# label_encoder object knows how to understand word labels.
label_encoder = preprocessing.LabelEncoder()

# Encode labels in column 'ExterQual'.
combined_data['ExterQual']= label_encoder.fit_transform(combined_data['ExterQual'])

# Encode labels in column 'ExterCond'.
combined_data['ExterCond']= label_encoder.fit_transform(combined_data['ExterCond'])

# Encode labels in column 'FireplaceQu'.
combined_data['FireplaceQu']= label_encoder.fit_transform(combined_data['FireplaceQu'])

# Encode labels in column 'BsmtQual'
combined_data['BsmtQual']= label_encoder.fit_transform(combined_data['BsmtQual'])

# Encode labels in column 'BsmtCond'
combined_data['BsmtCond']= label_encoder.fit_transform(combined_data['BsmtCond'])


# Encode labels in column 'GarageQual'
combined_data['GarageQual']= label_encoder.fit_transform(combined_data['GarageQual'])

# Encode labels in column 'GarageCond'
combined_data['GarageCond']= label_encoder.fit_transform(combined_data['GarageCond'])


# Encode labels in column 'KitchenQual'
combined_data['KitchenQual']= label_encoder.fit_transform(combined_data['KitchenQual'])


# Encode labels in column 'BsmtCond'
combined_data['BsmtCond']= label_encoder.fit_transform(combined_data['BsmtCond'])


##Reference: https://www.geeksforgeeks.org/ml-label-encoding-of-datasets-in-python/
```

```
# Adding Total Square Feet variable to the dataset

combined_data['TotalSF'] = combined_data['1stFlrSF'] + combined_data['2ndFlrSF']+combined_data['GrLivArea'] + combined_data["TotalBsmtSF"]
```

- **Log transformation**

As mentioned in the “Data Exploration” section, the distribution of the “SalePrice” variable is skewed to the right, in order to normalize the distribution, log transformation method is used. After log transformation, the distribution of the “SalePrice” turned to a normal.



```
#Removing Skewness
train_data["SalePrice"] = np.log1p(train_data["SalePrice"])

from scipy.stats import norm
sns.distplot(train_data['SalePrice'] , fit=norm)
plt.title('Density plot of train_data after log transformation')

print("Skewness: %f" % train_data['SalePrice'].skew())
print("Kurtosis: %f" % train_data['SalePrice'].kurt())
```

```
/usr/local/lib/python3.9/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed
in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axe
s-level function for histograms).
  warnings.warn(msg, FutureWarning)
Skewness: 0.123637
Kurtosis: 0.803323
```

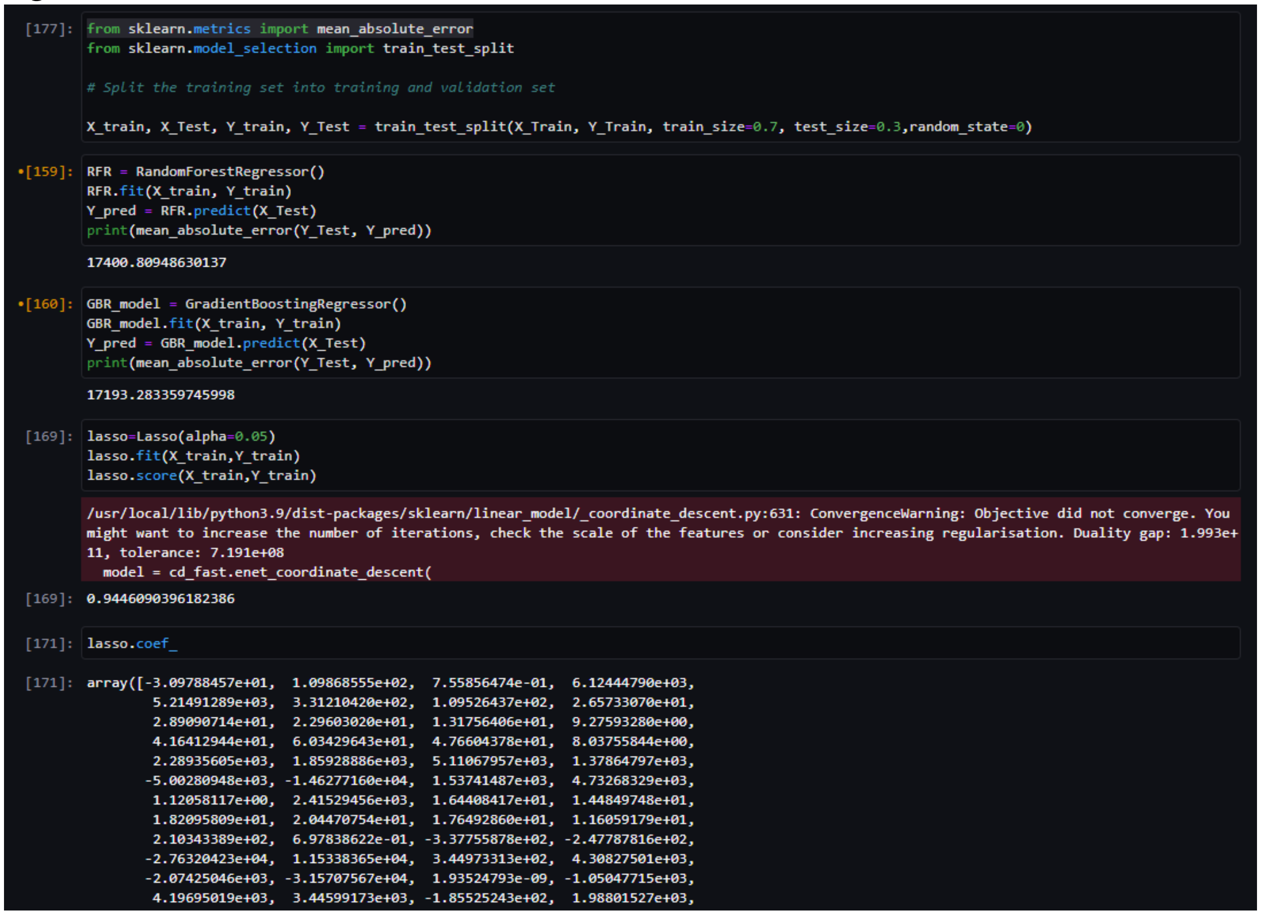
Lastly, I changed "MSSubClass", "OverallCond", “YrSold”, “MoSold” variables into categorical variables.

Regression Models and Comparison with the Benchmark model

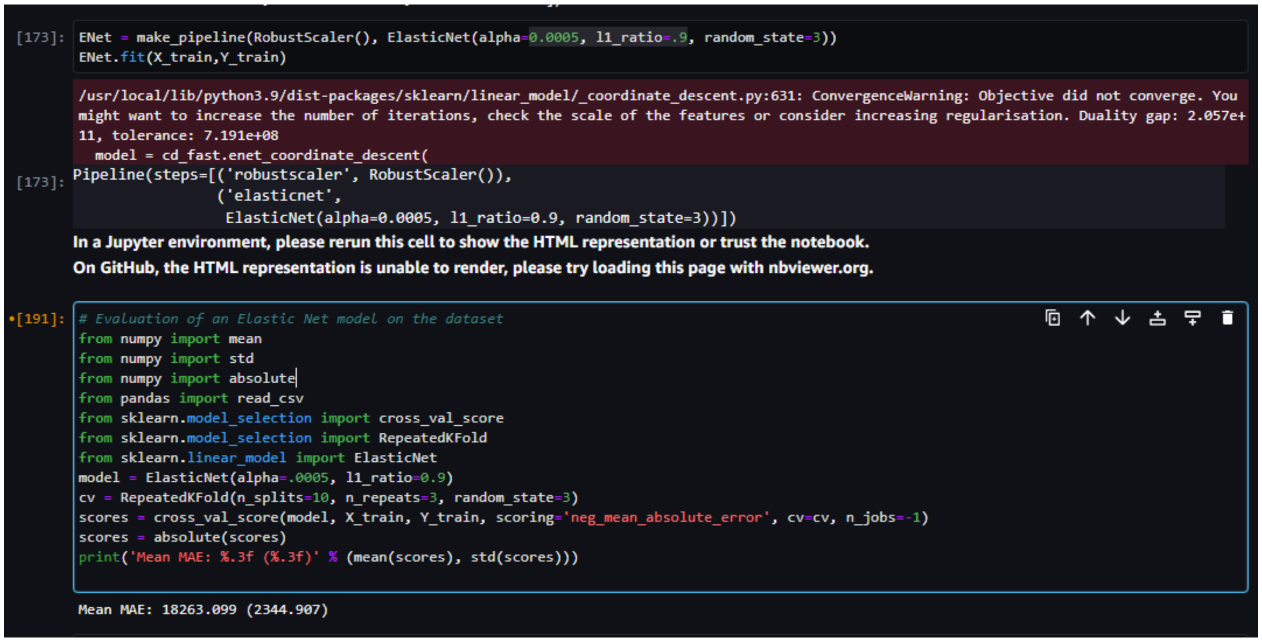
As mentioned in the “Algorithms and Techniques” part, after data cleaning and feature engineering, combined dataset are divided into train and test set.

In the benchmark model, the authors - Winky K.O. Ho, Bo-Sin Tang & Siu Wai Wong (2021) mainly utilized SVM (Support Vector Machines), Random Forest and Gradient Boosting Machine models to predict the house prices. The authors concluded that in comparison with the SVM results, Random Forest and Gradient Boosting Machine were able to generate comparably accurate price estimations with lower prediction errors. As I mentioned above, I additionally used Lasso Regression and Elastic Net model in order to predict the house prices.

Same as the Benchmark model, Mean Absolute Error (MAE) is used as an evaluation metric for this project. We can see the results of the Lasso, Elastic Net, Gradient Boost. Random Forest regression models below:



```
[177]: from sklearn.metrics import mean_absolute_error
       from sklearn.model_selection import train_test_split

       # Split the training set into training and validation set

       X_train, X_Test, Y_train, Y_Test = train_test_split(X_Train, Y_Train, train_size=0.7, test_size=0.3,random_state=0)

•[159]: RFR = RandomForestRegressor()
        RFR.fit(X_train, Y_train)
        Y_pred = RFR.predict(X_Test)
        print(mean_absolute_error(Y_Test, Y_pred))

        17400.80948630137

•[160]: GBR_model = GradientBoostingRegressor()
        GBR_model.fit(X_train, Y_train)
        Y_pred = GBR_model.predict(X_Test)
        print(mean_absolute_error(Y_Test, Y_pred))

        17193.283359745998

[169]: lasso=Lasso(alpha=0.05)
       lasso.fit(X_train,Y_train)
       lasso.score(X_train,Y_train)

       /usr/local/lib/python3.9/dist-packages/sklearn/linear_model/_coordinate_descent.py:631: ConvergenceWarning: Objective did not converge. You
       might want to increase the number of iterations, check the scale of the features or consider increasing regularisation. Duality gap: 1.993e+
       11, tolerance: 7.191e+08
         model = cd_fast.enet_coordinate_descent(
[169]: 0.9446090396182386

[171]: lasso.coef_

[171]: array([-3.09788457e+01,  1.09868555e+02,  7.55856474e-01,  6.12444790e+03,
               5.21491289e+03,  3.31210420e+02,  1.09526437e+02,  2.65733070e+01,
               2.89090714e+01,  2.29603020e+01,  1.31756406e+01,  9.27593280e+00,
               4.16412944e+01,  6.03429643e+01,  4.76604378e+01,  8.03755844e+00,
               2.28935605e+03,  1.85928886e+03,  5.11067957e+03,  1.37864797e+03,
              -5.00280948e+03, -1.46277160e+04,  1.53741487e+03,  4.73268329e+03,
               1.12058117e+00,  2.41529456e+03,  1.64408417e+01,  1.44849748e+01,
               1.82095809e+01,  2.04470754e+01,  1.76492860e+01,  1.16059179e+01,
               2.10343389e+02,  6.97838622e-01, -3.37755878e+02, -2.47787816e+02,
              -2.76320423e+04,  1.15338365e+04,  3.44973313e+02,  4.30827501e+03,
              -2.07425046e+03, -3.15707567e+04,  1.93524793e-09, -1.05047715e+03,
               4.19695019e+03,  3.44599173e+03, -1.85525243e+02,  1.98801527e+03,
```



```
[173]: ENet = make_pipeline(RobustScaler(), ElasticNet(alpha=0.0005, l1_ratio=.9, random_state=3))
       ENet.fit(X_train,Y_train)

       /usr/local/lib/python3.9/dist-packages/sklearn/linear_model/_coordinate_descent.py:631: ConvergenceWarning: Objective did not converge. You
       might want to increase the number of iterations, check the scale of the features or consider increasing regularisation. Duality gap: 2.057e+
       11, tolerance: 7.191e+08
         model = cd_fast.enet_coordinate_descent(
[173]: Pipeline(steps=[('robustscaler', RobustScaler()),
                       ('elasticnet',
                        ElasticNet(alpha=0.0005, l1_ratio=0.9, random_state=3))])
```

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.

```
•[191]: # Evaluation of an Elastic Net model on the dataset
        from numpy import mean
        from numpy import std
        from numpy import absolute
        from pandas import read_csv
        from sklearn.model_selection import cross_val_score
        from sklearn.model_selection import RepeatedKFold
        from sklearn.linear_model import ElasticNet
        model = ElasticNet(alpha=.0005, l1_ratio=0.9)
        cv = RepeatedKFold(n_splits=10, n_repeats=3, random_state=3)
        scores = cross_val_score(model, X_train, Y_train, scoring='neg_mean_absolute_error', cv=cv, n_jobs=-1)
        scores = absolute(scores)
        print('Mean MAE: %.3f (%.3f)' % (mean(scores), std(scores)))

        Mean MAE: 18263.099 (2344.907)
```

According to the result of the regression models, Lasso model performed the best.

References

- https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques/data[1]
- Winky K.O. Ho, Bo-Sin Tang & Siu Wai Wong (2021) Predicting property prices with machine learning algorithms, Journal of Property Research, 38:1, 48-70, DOI: 10.1080/09599916.2020.1832558[2]
- https://www.geeksforgeeks.org/ml-label-encoding-of-datasets-in-python/
- https://www.geeksforgeeks.org/implementation-of-lasso-regression-from-scratch-using-python/
- https://www.geeksforgeeks.org/implementation-of-lasso-ridge-and-elastic-net/?ref=lbp