MMAI Assignment 3 - Housing Prices ML Interpretation

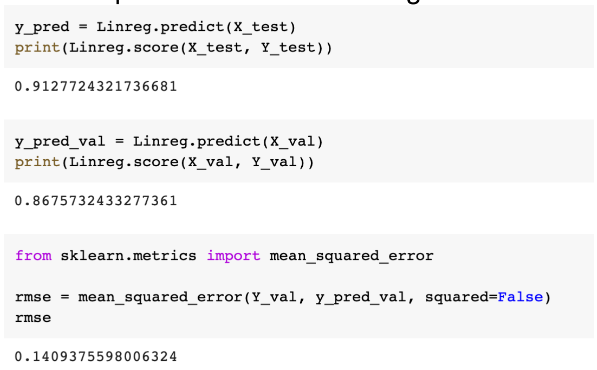
Intro:

In this project, after doing an extensive EDA and data wrangling, I fit two models: linear regression and XGBoost. I have also used hyperparameter tuning (grid search approach) to try different selections.
I have split the data to predict the accuracy of the learning algorithms. The score calculated:

	Prediction on X_Val	Prediction on X_test	RMSE
Linear Regression	0.86	0.91	0.14
XGBoost	0.89	0.90	0.12

Table 1: Accuracy Scores of Learning Algorithms

An example of the code used to get the values in Table 1:



```
y_pred = Linreg.predict(X_test)
print(Linreg.score(X_test, Y_test))
```

```
0.9127724321736681
```

```
y_pred_val = Linreg.predict(X_val)
print(Linreg.score(X_val, Y_val))
```

```
0.8675732433277361
```

```
from sklearn.metrics import mean_squared_error

rmse = mean_squared_error(Y_val, y_pred_val, squared=False)
rmse
```

```
0.1409375598006324
```

Figure 1: Example of code used to calculate accuracies shown in table 1

Note that we have RMSE in table 1. Root mean square error is a common way that is used for evaluating the quality of predictions. It shows how far predictions fall from measured true values**.**

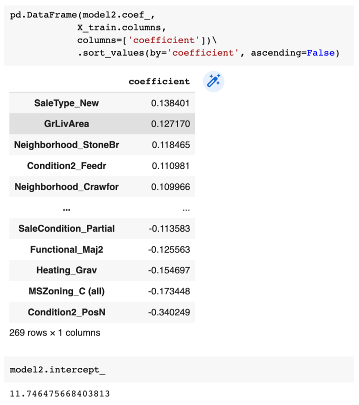
Linear Model (Linear Regression):

The Linear regression model: $Y = a + b_1X_1 + b_2X_2 + \ldots + b_mX_m$
Where Y is the prediction of the target and Xj (1<j<m) is the feature in the dataset. The b values (coefficients) can show the **sensitivity of the prediction** to each value of a feature. (Fig 1 shows the coefficients).
In order to interpret the linear regression model, we use the coefficients to explain the positive or negative correlation between each independent variable. A positive value shows that as the value of a feature increases, the mean of the dependent feature tends to increase. Also, how much the mean of the dependent features might change per one unit change of each independent feature. The bias, a, is a value of the target if all the feature values are zero. (Fig 2 shows the intercept).

1- Coefficient and Intercept Interpretation

As you can see in the figure 2, **SaleType_New, GrLivArea, Neighborhood_StoneBr** are features that have positive impact on the target feature (SalePrice) and **Condition2_PosN**, **MSZoning_C (all), Heating_Grav** have negative impact. As you can see the coefficient values are 0<x<1, this is due to **log transformation** that was implemented on our data. For example, let's take **SaleType_New** feature. You can also see that **GrLivArea** is one the top features in terms of coefficients too which was shown in our EDA process.

In terms of the intercept, we have calculated 11.74 which would be $e^{11.74}$ =125,492, which means if X equals zero, intercept is the expected mean value of Y at that value. Also, note that we have used dummy variables, which uses values of 0 for reference group and 1 for comparison group. So, intercept is the expected mean value when X=0, mean value for the reference group.



```
pd.DataFrame(model2.coef_,
             X_train.columns,
             columns=['coefficient'])\
             .sort_values(by='coefficient', ascending=False)
```

	coefficient
SaleType_New	0.138401
GrLivArea	0.127170
Neighborhood_StoneBr	0.118465
Condition2_Feedr	0.110981
Neighborhood_Crawfor	0.109966
...	...
SaleCondition_Partial	-0.113583
Functional_Maj2	-0.125563
Heating_Grav	-0.154697
MSZoning_C (all)	-0.173448
Condition2_PosN	-0.340249

269 rows × 1 columns

```
model2.intercept_
```

```
11.746475668403813
```

Figure 2: Dataframe of the features and the coefficients of each feature and the intercept value of the linear regression

2- Limitation:

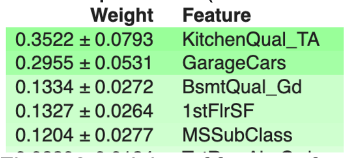
Using coefficients helps us to interpret the equations and explain the positive or negative correlation however, it does not tell you how well our linear regression model fits the data. In order to do that, we need to evaluate RMSE and R squared.

Non-linear Model (XGBoost):

Feature Permutation:

1- Method's Intro

In feature permutation we randomly shuffle a single column of the validation set and leaving the other columns to see how this change would affect our predictions' accuracy. We have fit our model on the split data (validation set, train set, and test set). The result of feature permutation is:



Weight	Feature
0.3522 ± 0.0793	KitchenQual_TA
0.2955 ± 0.0531	GarageCars
0.1334 ± 0.0272	BsmtQual_Gd
0.1327 ± 0.0264	1stFlrSF
0.1204 ± 0.0277	MSSubClass

Figure 3: weights of feature after feature permutation

2- Interpretation

The values on top of the table in Figure 3, seem to be the most important features and the numbers represent how much the model's error changes when shuffling a column. We might see negative values which mean that they did not matter. The most important features according to the figure 3, are **KitchenQual_TA, GarageCars, BsmtQual_Gd, 1stFlrSF, MSSubClass.** The weights show how much it would influence the model's error when, for example, 1stFlrSF information is shuffled.

3- Feature Interaction

In order to measure the importance of interaction between features, we need to understand that when we shuffle a feature, not only it destroys its information but also it destroys its interaction with other features. So, for example, if **GarageCars** has a correlation with another feature, let's say,

GarageArea, when **GarageCars** is shuffled not only it has a great impact on the model's error but it has killed its interaction with **GarageArea** feature.

4- Limitation

As mentioned above, feature permutation uses feature shuffling which creates randomness to measurement, and repetitive permutation lead to different results.
Also, feature permutation is connected to the model's error. So, in some cases where you just want to know how much the model's output would change when manipulating the features, but not how much the model performance decreases, you would not use feature permutation approach to interpret the model.

ICE Plot:

1- Method's Intro

Individual Conditional Expectation (ICE) shows the dependence of the prediction on a feature for each instance separately, as opposed to PDP plot for average effect of a feature that does not focus on a specific instance.

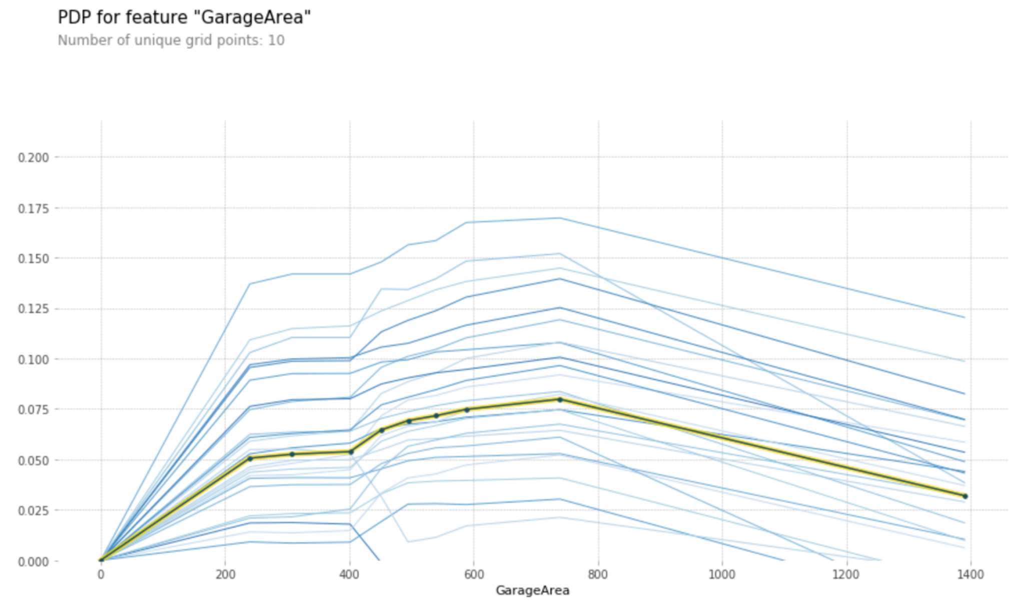


Figure 4: Univariate ICE plot

2- Interpretation

In Fig 4, we can see the relationships between subsets of the instances as well as differences in how individual instances behave. Fig 4 shows an instance of interest (**GarageArea**) and most of the instances follow shape of the curve; however, on the x-axis between 400-600 we can see a drop of a small subset at y-axis value of 0.050 & 0.020 instead of increasing.

3- Feature Interaction

A line in figure 4 is computed by keeping all the other features the same, and changing the target feature (**SalePrice**). As we mentioned, PDP plot for average effect of a feature on prediction, this can be only useful if the interaction between the '**GarageArea'** (our feature of intereset) and other features is weak. If the feature interaction is not weak, ICE can provide much more useful information.

4- Limitations
ICE plots can interpret only one feature in one plot. If many features are drawn, it would be overcrowded. Another problem with ICE plots is potential of high correlations between features because for example two feature with high correlation can lead to difficulty of knowing of how much of the predictive influence is because of either feature.

Shapely Values:

1- Method’s Intro
In this method, we want to understand the contribution of the features to the change in predictions. To do this, we re-run the model to calculate the prediction of every situation where some features hold their current values and some features have their average values. Calculating the contribution of features using shapely values brings some benefits for example, if a feature does not change, its contribution to prediction is zero.
Since our ML model is decision tree based, to obtain Shapley values we will use TreeShap.

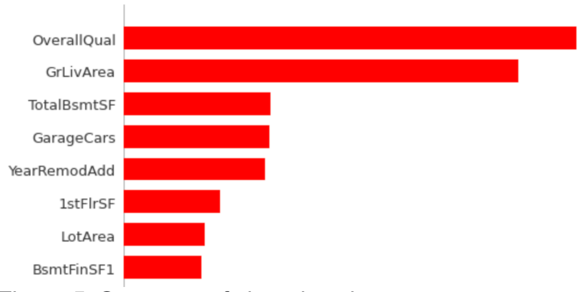


Figure 5: Summary of shapely values

2- Interpretation
Shapley value calculated in figure 5, are the average contribution of features which are predicting the target in different situations. In this chart, we can understand the feature importance according the average SHAP values magnitudes. The first 5 features are: ‘**OverallQual’**, ‘**GrLivArea’**, ‘**TotalBsmtSF’**, ‘**GarageCars’**, ‘**YearRemodAdd’**. According to this method, these features would have the highest contribution in the change in predictions.

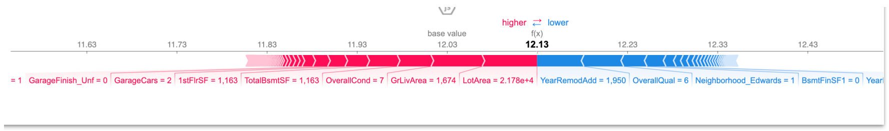


Figure 6: force plot for positive and negative samples

As we can see in fig 6, the red features have shapely values that are pushing the prediction higher than the base value are shown in red. In blue we see features that they push prediction towards lower than the base value.

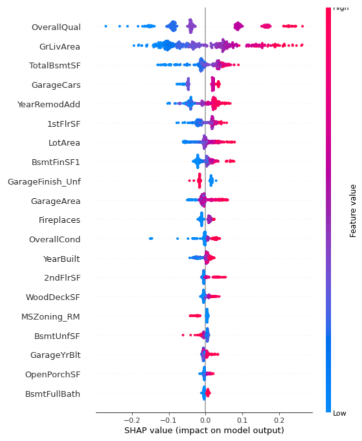


Figure 7: summary plot of Shapely values

In the figure 7, we can get information about the feature importance as well as the impact of the Shapley value by changing the value of each feature. ‘**OverQual**’ being the most important and ‘**BsmtFullBath’** being the least important in this plot. Another observation from this plot is that the first three features generate the most impact on the predictions due to their Shapely values, meaning their Shapley values take a larger range, some are very high or very low.

3- Feature Interaction

```
shap.dependence_plot(ind='GrLivArea', interaction_index='TotalBsmtSF',
                     shap_values=shap_values,
                     features=X_test,
                     display_features=X_test)
```

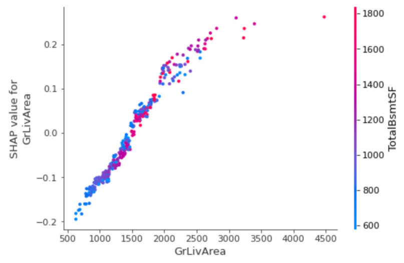


Figure 8: dependence plot of two features: ‘GrLivArea’, ‘TotalBsmtSF’

In the figure 8, it shows the feature value vs the SHAP values the feature. Here the features of interests are ‘**GrLivArea’**, ‘**TotalBsmtSF’**. In this graph we can also observe the interaction effects present in the features. As it has been observed before in our EDA and the results after model fitting, these two features of interest have high correlation which can also be seen here.

4- Limitations

Shapely values can be used to interpret why any two predictions are different; they are not capable enough to explain the model as a whole and predict how varying inputs would affect the prediction. Moreover, this approach can be computationally very complex and time consuming when there are many features; for example, in this case of housing prices project. Lastly, high correlation between features, can cause unrealistic combinations of feature values to be considered.

Conclusion

In this project we used Linear Regression model and XGBoost to predict housing prices. In order to get more accurate results; various methods have been used for example hyperparameter tuning using Grid search and etc. As you can see in table 1, we can compare the scores calculated for each model and RMSE and R^2 have been used to evaluate accuracy of the models. (please refer to table 1 to see the results). In order to interpret the liner regression model, we only looked at the coefficients and the intercept. The results observed have some similarities with the XGBoost model explaining results; for example, **GrLivArea** feature, was observed in both models. To interpret XGBoost, we used 3 different approaches: **Feature Permutation**, **ICE plot**, and **Shapely values**. Feature permutation focuses on error which it can lead to overfitting. ICE plot shows feature dependence on prediction for each instance separately and shapely values only focuses on the contribution of features to change in prediction. The results calculated from these methods might contradict on some level but they also have some similarities. For example, according to feature permutation, the most important feature is **KitchenQual_TA** but according to Shapely values, **OveralQual** would be the most important feature. But as we think about **OveralQual**, we realize that **KitchenQual_TA** feature is kind of included in **OverQual** feature. Also, both methods have **1stFlrSF** feature in their top 6 features.