NAME OF THE PROJECT

Housing Project

Submitted by:

Chandan Kumar

ACKNOWLEDGMENT:

This includes mentioning of all the references, research papers, data sources, professionals and other resources that helped me and guided me in completion project.

INTRODUCTION

1. Problem statement

The house buyer, a client that wants to buy their dream home. They have some locations in mind. Now, the client wants to know if the house price matches the value. With the application, they can understand which features are influence the final price. If the final price matches the value predicted by the application the can ensure they are getting a fair price.

Explore Dataset:

```
df_train = pd.read_csv('Documents/train.csv')
df_test = pd.read_csv('Documents/test.csv')
```

```
df_train.head(10)
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC	Fence	MiscFeature	MiscVal
0	127	120	RL	NaN	4928	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0
1	889	20	RL	95.0	15865	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0
2	793	60	RL	92.0	9920	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0
3	110	20	RL	105.0	11751	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	MnPrv	NaN	0
4	422	20	RL	NaN	16635	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0
5	1197	60	RL	58.0	14054	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0
6	561	20	RL	NaN	11341	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0
7	1041	20	RL	88.0	13125	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	GdPrv	NaN	0
8	503	20	RL	70.0	9170	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	GdPrv	Shed	400
9	576	50	RL	80.0	8480	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	0

10 rows × 81 columns

The dataset has 81 columns:

```
df_train.dtypes
```

```
Id                int64
MSSubClass        int64
MSZoning         object
LotFrontage     float64
LotArea           int64
                  ...
MoSold            int64
YrSold            int64
SaleType         object
SaleCondition    object
SalePrice         int64
Length: 81, dtype: object
```

Data Pre-Processing

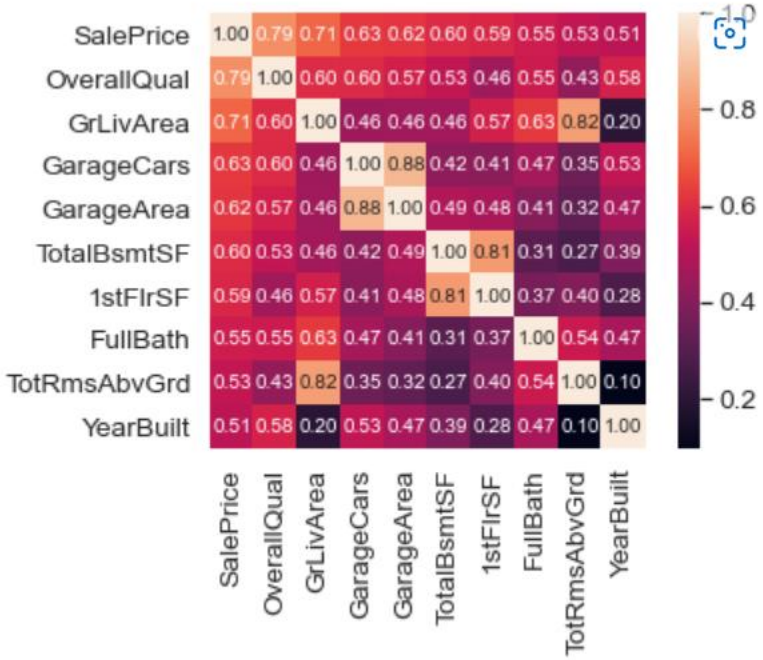
```
df_train.isnull().sum()
```

```
Id                  0
MSSubClass          0
MSZoning            0
LotFrontage       214
LotArea             0
                   ...
MoSold              0
YrSold              0
SaleType            0
SaleCondition       0
SalePrice           0
Length: 81, dtype: int64
```

Exploratory Data Analysis

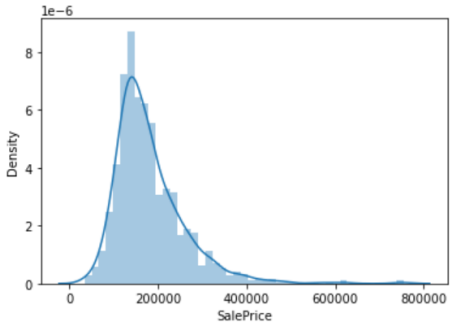
As the purpose of this experiment is to identify patterns that can yield to Housing Price, I will be focusing mainly on the housing price portion of the dataset for the exploratory analysis.

```
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, fr
plt.show()
```

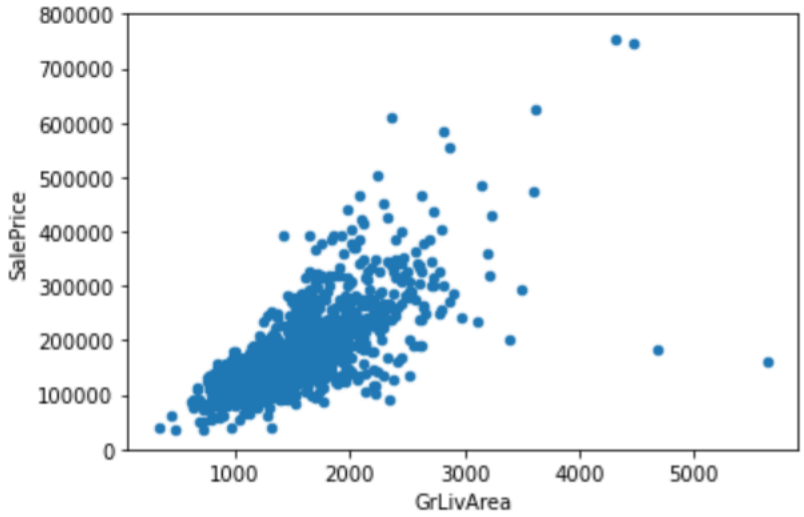



```
sns.distplot(df_train['SalePrice']);
```

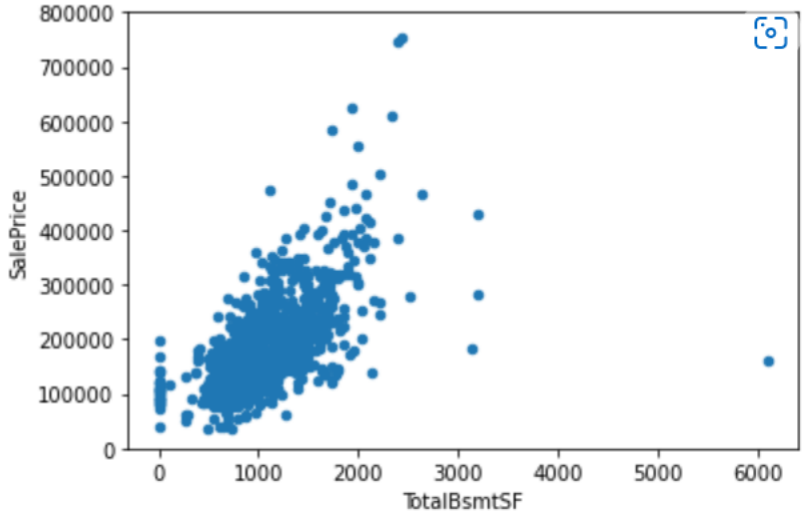
```
C:\Users\Administrator\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated fun
ction and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with sim
ilar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



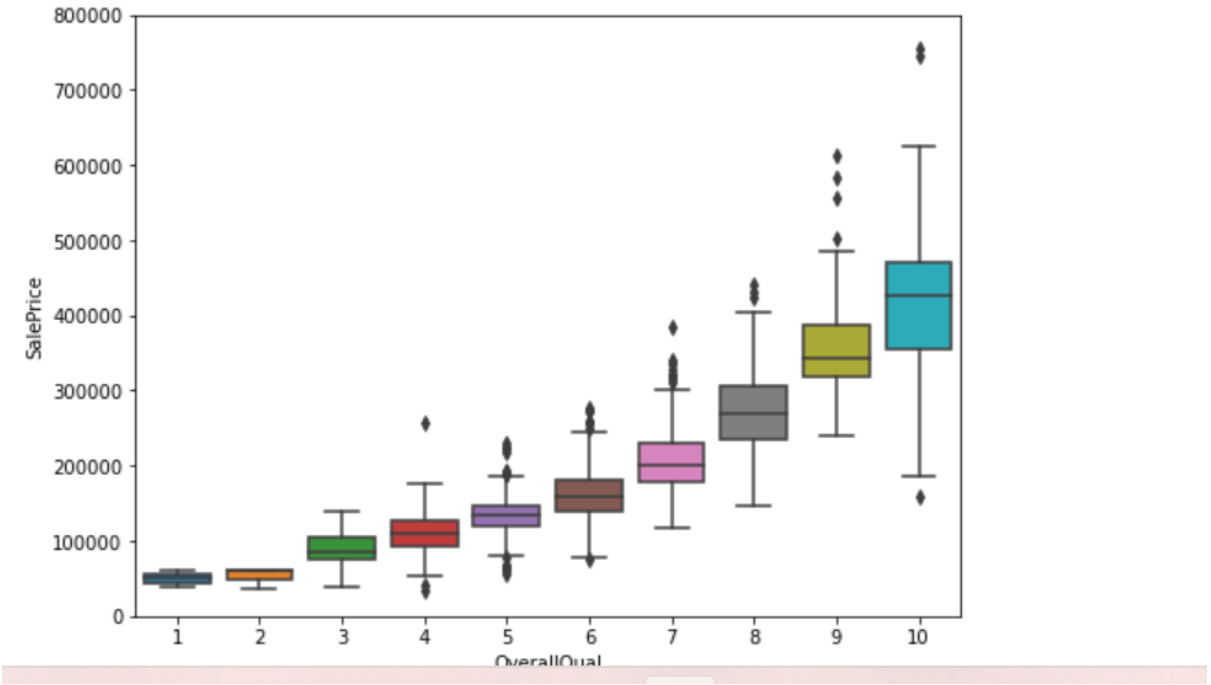
```
var = 'GrLivArea'
data = pd.concat([df_train['SalePrice'], df_train[var]], axis=1)
data.plot.scatter(x=var, y='SalePrice', ylim=(0,800000));
```




```
var = 'TotalBsmtSF'
data = pd.concat([df_train['SalePrice'], df_train[var]], axis=1)
data.plot.scatter(x=var, y='SalePrice', ylim=(0,800000));
```



```
var = 'OverallQual'
data = pd.concat([df_train['SalePrice'], df_train[var]], axis=1)
f, ax = plt.subplots(figsize=(8, 6))
fig = sns.boxplot(x=var, y="SalePrice", data=data)
fig.axis(ymin=0, ymax=800000);
```



Summary of EDA

- 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.
- '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

```
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
```

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

```
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'])
```

```
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
```

```
ml_algo_compare
```

	MLAlgo Name	MLAlgo Parameters	MLAlgo Train MAE	MLAlgo Test MAE	MLAlgo Time
0	LinearRegression	{'copy_X': True, 'fit_intercept': True, 'n_job...	-19178.132407	-22050.500239	0.020945
1	DecisionTreeRegressor	{'ccp_alpha': 0.0, 'criterion': 'squared_error...	0.0	-26863.411597	0.021652
2	RandomForestRegressor	{'bootstrap': True, 'ccp_alpha': 0.0, 'criteri...	-6939.957799	-18147.739306	1.016005
3	SVR	{'C': 1.0, 'cache_size': 200, 'coef0': 0.0, 'd...	-55247.248512	-55411.59805	0.052072
4	XGBRegressor	{'objective': 'reg:squarederror', 'base_score'...	-274.936169	-19114.910603	0.270773
5	KNeighborsRegressor	{'algorithm': 'auto', 'leaf_size': 30, 'metric...	-19727.986239	-23995.269408	0.010414

```
ml_algo_predict.sample(10)
```

	Actual Sales	LinearRegression	DecisionTreeRegressor	RandomForestRegressor	SVR	XGBRegressor	KNeighborsRegressor
2	269790	239467.370078	269790.0	251516.50	164030.329365	266799.375000	248318.0
84	159500	145221.630046	159500.0	160313.50	163944.164335	160533.781250	156100.0
911	270000	263107.232066	270000.0	254189.91	163962.036582	269811.812500	241300.0
325	167500	208984.746872	167500.0	178672.90	163944.262416	167648.453125	155500.0
203	140000	133288.144226	140000.0	140767.37	163901.768017	139371.046875	134000.0
941	125500	107463.244850	125500.0	124768.00	163898.668032	124402.281250	123000.0
548	85000	111240.854811	85000.0	90754.50	163912.636219	85095.054688	100740.0
273	320000	326876.035638	320000.0	310224.10	164007.372575	319804.531250	320456.0
460	137900	151515.802150	137900.0	148943.59	163940.912715	137513.015625	143860.0
652	125000	78803.394260	125000.0	125171.51	163944.987450	124777.945312	188581.0

```
dtmodel = XGBRegressor()
dtmodel.fit(trn_data, targets)
```

```
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, ...)
```

```
from sklearn.metrics import mean_absolute_error, r2_score
trn_p = dtmodel.predict(trn_data)
tst_ = dtmodel.predict(tst_data)
```

```
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
```

We philosophied 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.