Predicting Housing Sale price

Poornima
Micky Kumar
Ayswarya Venkatraman Kandasamy

If your not going to put money in real estate… Where else

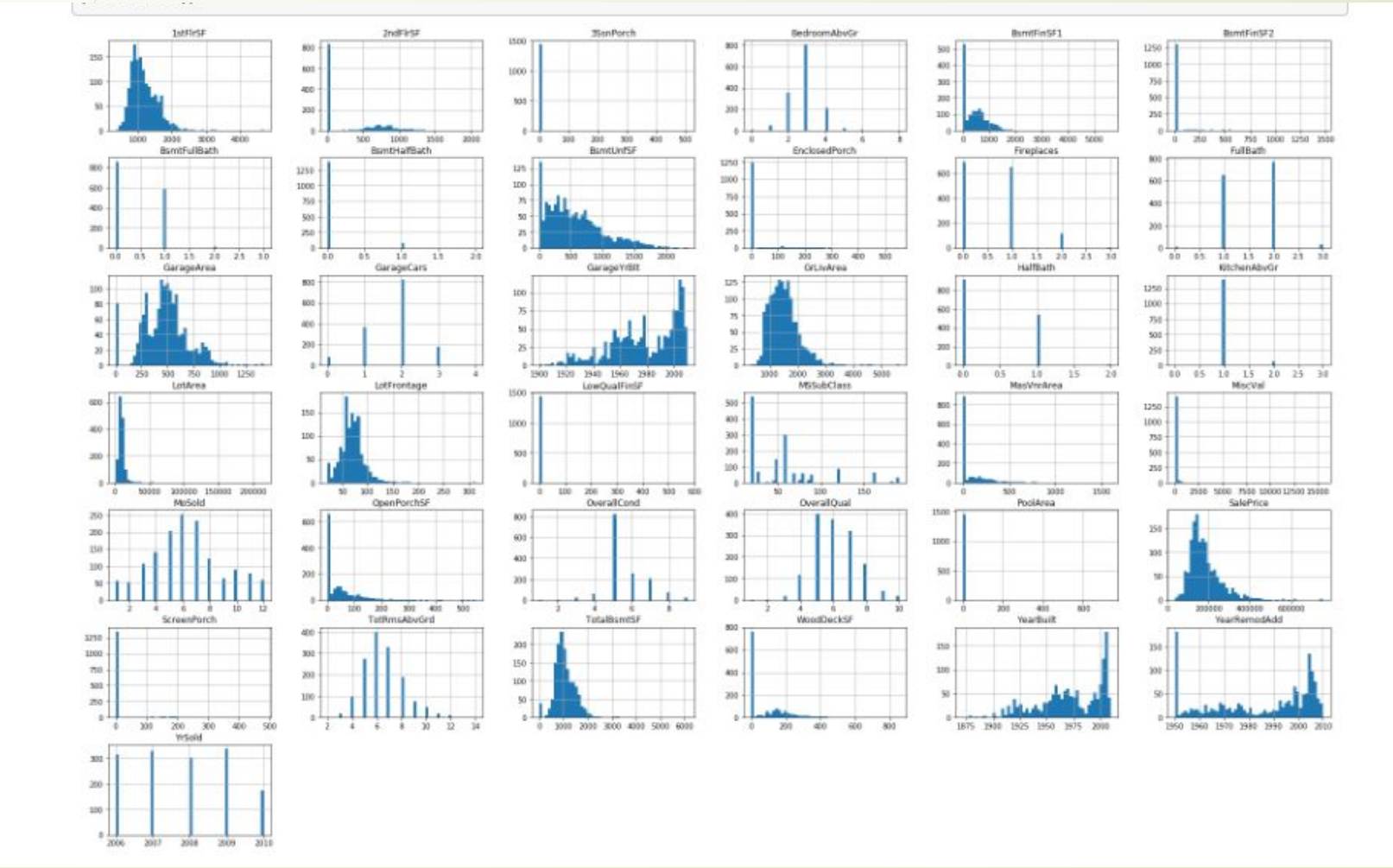
Objective:

- Develop and evaluate the performance and the predictive model to train and test on data collected from house sale price

Data Process:

- Explore data
- 79 explanatory variables
- Feature selection
- Cleaning data
- Prepare data for model

Explore

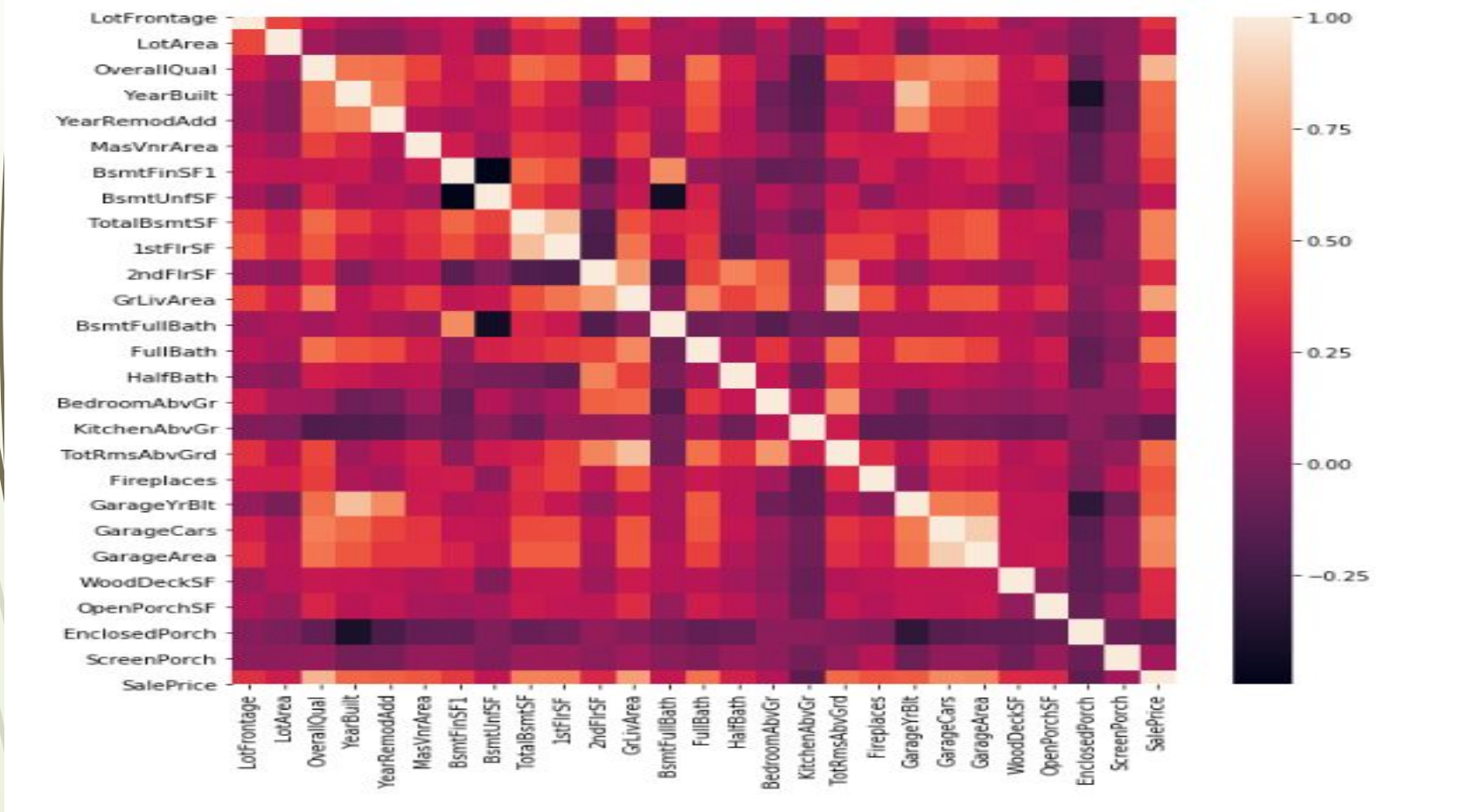

1stFlrSF
2ndFlrSF
3SsnPorch
BedroomAbvGr
BsmtFinSF1
BsmtFinSF2
BsmtFullBath
BsmtHalfBath
BsmtUnfSF
EnclosedPorch
Fireplaces
FullBath
GarageArea
GarageCars
GarageYrBlt
GrLivArea
HalfBath
KitchenAbvGr
LotArea
LotFrontage
LowQualFinSF
MSSubClass
MasVnrArea
MiscVal
MoSold
OpenPorchSF
OverallCond
OverallQual
PoolArea
SalePrice
ScreenPorch
TotRmsAbvGrd
TotalBsmtSF
WoodDeckSF
YearBuilt
YearRemodAdd
YrSold

Missing values

```
LotFrontage     0.184253
MasVnrArea      0.010388
BsmtFinSF1      0.000686
BsmtFinSF2      0.000686
BsmtUnfSF       0.000686
TotalBsmtSF     0.000686
BsmtFullBath    0.001373
BsmtHalfBath    0.001373
GarageYrBlt     0.056481
GarageCars      0.000686
GarageArea      0.000686
dtype: float64
```

```
LotFrontage     0.215654
MasVnrArea      0.005510
GarageYrBlt     0.058738
dtype: float64
```

Correlation Map:



Feature Engineering :

```
def addextraattributes(df):
    now=dt.datetime.now()
    df['age_of_house']=now.year-df['YearBuilt']
    df['remodd_age']=now.year-df['YearRemodAdd']
    df['year_sold']=(df['YrSold'].astype(str)+(df['MoSold'].astype(str)).str.zfill(2)).astype(int)
    df['totalsqft']= df['TotalBsmtSF']+df['1stFlrSF']+df['2ndFlrSF']
    df['total_washroom']=df['BsmtFullBath']+df['BsmtHalfBath']+df['FullBath']+df['HalfBath']
    return df
```

```
])

ordPipeline=Pipeline([
    ('imp',SimpleImputer(missing_values=np.nan,strategy='most_frequent'))
])

fullPipeline=ColumnTransformer([
    ('num',numPipeline,numcols),
    ("ord",ordPipeline,ordcols),
    ("cat",ce.BinaryEncoder(),catcols)
])
analysiscols=catcols+numcols+ordcols
```

Regression Algorithms:

	Cross Validation Score(CV=5)
Linear Regression	0.82248439, 0.80204839, 0.85063114, 0.85845217, 0.64173928
Lasso	0.82283526, 0.80250871, 0.87054813, 0.85880556, 0.64167979
Elastic Net	0.87630322, 0.8328449, 0.83531939, 0.85993959, 0.67800232
Kneighbors Regressor	0.7948717, 0.73540321, 0.79503801, 0.80827136, 0.69184637
Gaussian NB	0.00819672, 0.01644737, 0.01485149, 0.01526718, 0
SVR	-0.07021949, -0.06052138, -0.05574727, -0.01534855, -0.05494959
Gradient Boosting Regressor	0.90443464, 0.83201152, 0.90386332, 0.90941942, 0.89522425
Random Forest Regressor	0.87040612, 0.83648642, 0.88337562, 0.83135065, 0.83663928

Random Forest Regression:

```
rf_param=[{'n_estimators': [500,600,700,800], 'max_features':['log2','sqrt']}]
forest_reg = RandomForestRegressor(random_state=42)
grid_search = GridSearchCV(forest_reg,cv=5,param_grid=rf_param,
                           scoring='neg_mean_squared_error', return_train_score=True)
grid_search.fit(Xprep,Y)
model=grid_search.best_estimator_
cvresult=pd.DataFrame(grid_search.cv_results_)
cvresult['sqrt_mean_test_score']=np.sqrt(-cvresult['mean_test_score'])
cvresult['sqrt_mean_train_score']=np.sqrt(-cvresult['mean_train_score'])
cvresult[['params','sqrt_mean_test_score','sqrt_mean_train_score']]
```

	params	sqrt_mean_test_score	sqrt_mean_train_score
0	{'max_features': 'log2', 'n_estimators': 500}	31622.053875	11666.357556
1	{'max_features': 'log2', 'n_estimators': 600}	31624.444163	11670.426755
2	{'max_features': 'log2', 'n_estimators': 700}	31672.911783	11642.540940
3	{'max_features': 'log2', 'n_estimators': 800}	31595.607796	11645.450286
4	{'max_features': 'sqrt', 'n_estimators': 500}	29814.492873	11024.313556
5	{'max_features': 'sqrt', 'n_estimators': 600}	29840.836432	11060.550528
6	{'max_features': 'sqrt', 'n_estimators': 700}	29868.849445	11083.940960
7	{'max_features': 'sqrt', 'n_estimators': 800}	29880.494825	11080.369627

Gradient Boosting Regressor:

```
gb_param=[{'alpha':[.9],'learning_rate':[.09,.05,.025,.001],'n_estimators': [2000,2200],
          'min_impurity_decrease':[1e-5,1e-6],
          'max_features':[2,3,4,'log2'],'warm_start':[True]}]
gb_reg = GradientBoostingRegressor(random_state=42)
grid_search = GridSearchCV(gb_reg,cv=5,param_grid=gb_param,
                           scoring='neg_mean_squared_error', return_train_score=True)
grid_search.fit(Xprep,Y)
model=grid_search.best_estimator_
cvresult=pd.DataFrame(grid_search.cv_results_)
cvresult['sqrt_mean_test_score']=np.sqrt(-cvresult['mean_test_score'])
cvresult['sqrt_mean_train_score']=np.sqrt(-cvresult['mean_train_score'])
cvresult[['params','sqrt_mean_test_score','sqrt_mean_train_score']]
```

	params	sqrt_mean_test_score	sqrt_mean_train_score
0	{'alpha': 0.9, 'learning_rate': 0.09, 'max_fea...	28135.947331	7639.025544
1	{'alpha': 0.9, 'learning_rate': 0.09, 'max_fea...	28117.513388	7162.612707
2	{'alpha': 0.9, 'learning_rate': 0.09, 'max_fea...	28135.947331	7639.025544
3	{'alpha': 0.9, 'learning_rate': 0.09, 'max_fea...	28117.513388	7162.612707
4	{'alpha': 0.9, 'learning_rate': 0.09, 'max_fea...	28109.575188	5819.091449
...	...	...	...
59	{'alpha': 0.9, 'learning_rate': 0.001, 'max_fe...	44026.985305	40699.620143
60	{'alpha': 0.9, 'learning_rate': 0.001, 'max_fe...	41293.590049	37479.274581
61	{'alpha': 0.9, 'learning_rate': 0.001, 'max_fe...	39757.361532	35561.658377
62	{'alpha': 0.9, 'learning_rate': 0.001, 'max_fe...	41293.590049	37479.274581
63	{'alpha': 0.9, 'learning_rate': 0.001, 'max_fe...	39757.361532	35561.658377

64 rows × 3 columns

Important Features:

feature_imp

	0	1
0	0.109298	PoolQC
1	0.067776	GarageCond
2	0.059033	Condition2
3	0.058505	PavedDrive
4	0.048888	RoofStyle
...	...	...
79	0.000000	MiscFeature
80	0.000000	HalfBath
81	0.000000	GrLivArea
82	0.000000	GarageYrBlt
83	0.000000	EnclosedPorch

84 rows × 2 columns

Results:

Id	SalePrice
1461	131365.596230
1462	162114.105864
1463	192895.900918
1464	193101.253133
1465	185989.945659
...	...
2915	77638.432540
2916	76997.317268
2917	154678.655844
2918	122948.220355
2919	221138.792230

1459 rows × 1 columns

Conclusion:

- Gradient Boosting Regressor provided best regression results compared to random forest regressor
- Kaggle results
- Predictive model like this would be very valuable for a real state agent & home buyers who could make use of the information provided in a daily basis