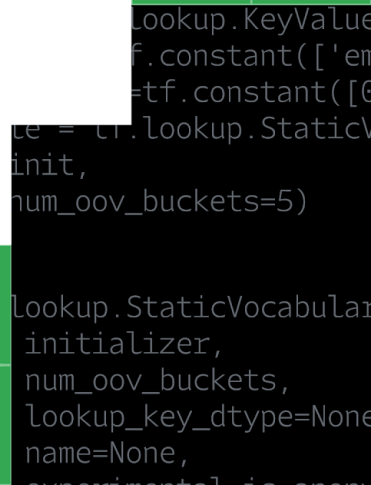
Google Developer Student Clubs

House price Prediction

ML/DL Basic

한유진



Code Review - Missing Value

df = df.dropna(axis=1, how='any')
: 결측치를 삭제하는 작업

```
1 df = df.dropna(axis=1, how='any')
```

isnull().sum()
: column별로 결측치의 개수를 계산하는 작업

```
1 df.isnull().sum()
```

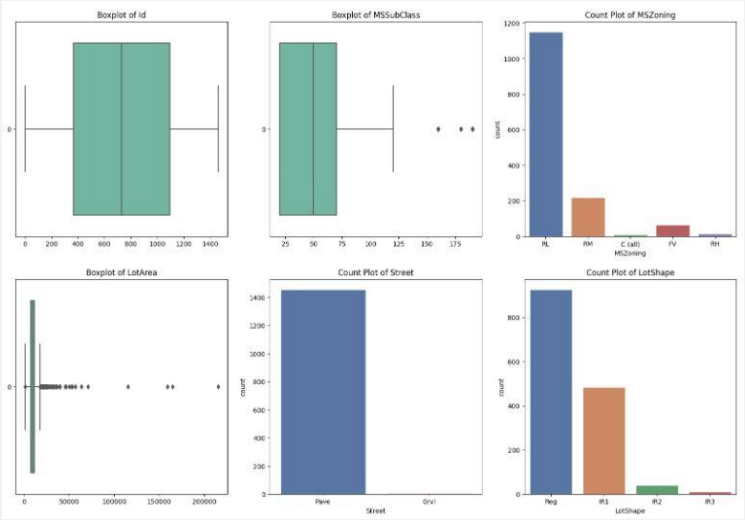
```
Id               0
MSSubClass       0
MSZoning         0
LotFrontage    259
LotArea          0
              ...
MoSold           0
YrSold           0
SaleType         0
SaleCondition    0
SalePrice        0
Length: 81, dtype: int64
```

```
1 df.isnull().sum()
```

```
Id               0
MSSubClass       0
MSZoning         0
LotArea          0
Street           0
                ..
MoSold           0
YrSold           0
SaleType         0
SaleCondition    0
SalePrice        0
Length: 62, dtype: int64
```

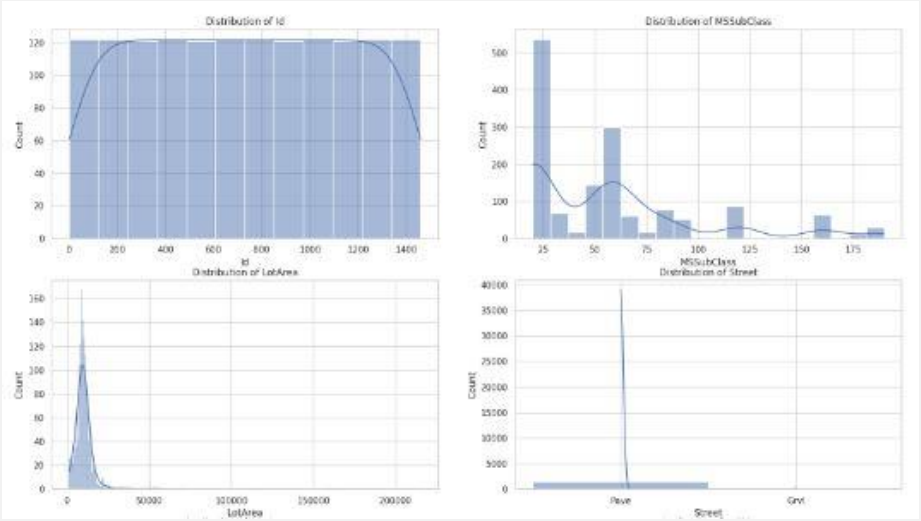
Code Review - Outlier

Column 데이터를 시각화하는 작업



수치형 데이터는 boxplot,

범주형 데이터는 countplot 생성



데이터의 분포 확인

Code Review - Outlier

```python
def remove_outliers(df, columns, threshold=1.5):
    for column in columns:
        Q1 = df[column].quantile(0.25)
        Q3 = df[column].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - threshold * IQR
        upper_bound = Q3 + threshold * IQR
        df = df[(df[column] >= lower_bound) & (df[column] <= upper_bound)]
    return df

columns_to_remove_outliers = ['MSSubClass','OverallQual','OverallCond','BsmtFinSF1',
removed_outlier = remove_outliers(ptr_df, columns_to_remove_outliers)
removed_outlier
```

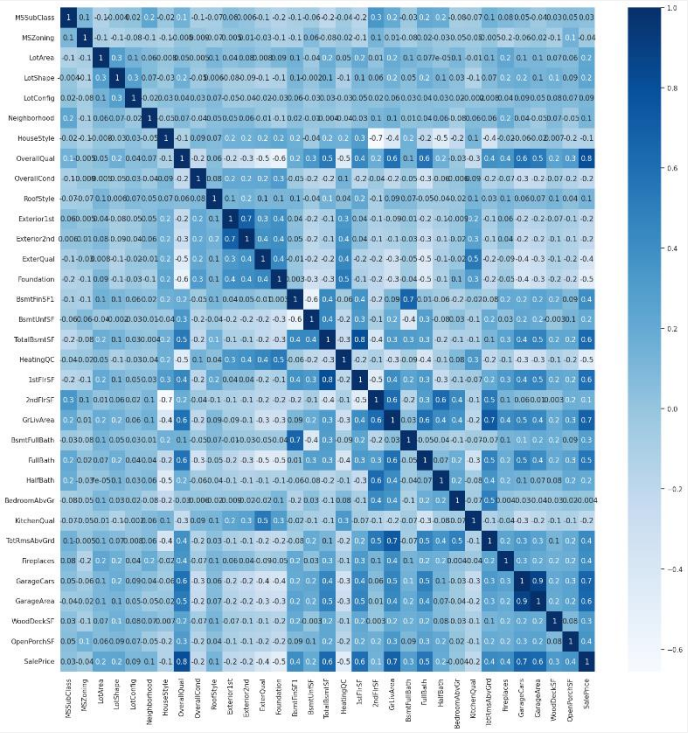
Outlier 제거
: IQR을 설정하여 이상치를 정의
(Q1 - threshold * IQR 보다 작거나
Q3 + threshold * IQR 보다 큰 값)

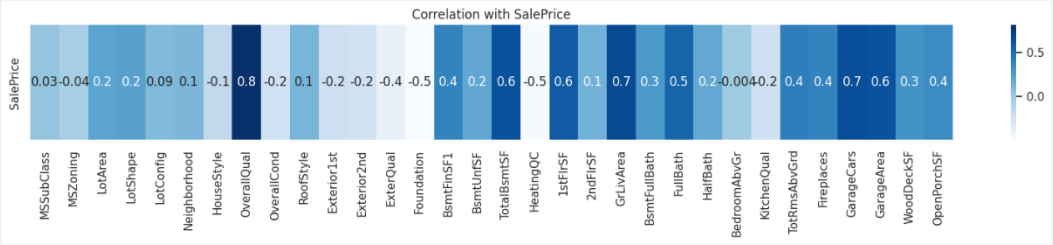
```python
def factorize_categorical_columns(column):
    if column.dtype == 'object':
        column_encoded, _ = pd.factorize(column)
        return column_encoded
    return column

# Apply factorize only to categorical columns
df_encoded = removed_outlier.apply(factorize_categorical_columns)
```

범주형 데이터를 정수로 인코딩
: pd.factorize 함수를 이용하여
데이터 유형이 object인 경우 범주형
데이터로 간주해 수치형 데이터로 변환

Code Review - Correlation Heatmap





Correlation analysis

: 두 변수간의 관계를 상관계수로 표현하는 작업

final_df.corr(): 기본적으로 pearson 상관계수 계산

Pearson 상관계수: -1과 1 사이의 값을 가짐

Code Review - VIF

VIF(Variance Inflation Factor)

: 분산 팽창 인수로 데이터들이 다중공선성이 있는지 확인하고 10이 넘으면 문제가 있음

다중공선성 - 독립변수들 간에 상관관계를 가짐

```
1 from statsmodels.stats.outliers_influence import variance_inflation_factor
2 vif=pd.DataFrame()
3 vif['VIF']=[variance_inflation_factor(sel_df,i) for i in range(sel_df.shape[1])]
4 vif['features']=sel_df.columns
```

Code Review - Applying ML Algorithm

Linear Regression 진행

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.linear_model import LinearRegression

model = LinearRegression()
model.fit(X_train, y_train)
```

```
test_pred=model.predict(X_test)
end_pred=pd.DataFrame(test_pred,index=df_test.index)
end_pred.columns=['SalePrice']
end_pred.to_csv('submission.csv',sep=',')
end_pred.head()
```

Performance Improvement

GBoost Regression

```
GBoost = GradientBoostingRegressor(n_estimators=3000, learning_rate=0.05,
                                   max_depth=4, max_features='sqrt',
                                   min_samples_leaf=15, min_samples_split=10,
                                   loss='huber', random_state =5)
```

```
GBoost.fit(train,y_train)
GB_train_pred = GBoost.predict(train)
GB_pred = np.expm1(GBoost.predict(test.values))
print(mse(y_train, GB_train_pred))
```

Performance Improvement

XGBoost Regression

```
model_xgb = xgb.XGBRegressor(colsample_bytree=0.4603, gamma=0.0468,
                             learning_rate=0.05, max_depth=3,
                             min_child_weight=1.7817, n_estimators=2200,
                             reg_alpha=0.4640, reg_lambda=0.8571,
                             subsample=0.5213, silent=1,
                             random_state =7, nthread = -1)
```

```
model_xgb.fit(train, y_train)
xgb_train_pred = model_xgb.predict(train)
xgb_pred = np.expm1(model_xgb.predict(test))
print(mse(y_train, xgb_train_pred))
```

Performance Improvement

LightGBM Regression

```
model_lgb = lgb.LGBMRegressor(objective='regression',num_leaves=5,
                              learning_rate=0.05, n_estimators=720,
                              max_bin = 55, bagging_fraction = 0.8,
                              bagging_freq = 5, feature_fraction = 0.2319,
                              feature_fraction_seed=9, bagging_seed=9,
                              min_data_in_leaf =6, min_sum_hessian_in_leaf = 11)
```

```
model_lgb.fit(train, y_train)
lgb_train_pred = model_lgb.predict(train)
lgb_pred = np.expm1(model_lgb.predict(test.values))
print(mse(y_train, lgb_train_pred))
```

Performance Improvement

Ensemble

```
ensemble1 = xgb_pred*0.25 + lgb_pred*0.25 + GB_pred*0.5
```

```
ensemble2 = xgb_pred*0.5 + lgb_pred*0.25 + GB_pred*0.25
```

```
ensemble3 = xgb_pred*0.25 + lgb_pred*0.5 + GB_pred*0.25
```

Performance Improvement

submission_ensemble3.csv Complete · now	0.12533
submission_ensemble2.csv Complete · 24s ago	0.12662
submission_ensemble1.csv Complete · 1m ago	0.12565
submission_xgb.csv Complete · 1m ago	0.13247
submission_lgb.csv Complete · 1m ago	0.12624
submission_GB.csv Complete · 2m ago	0.12897
submission (1).csv Complete · 7h ago	0.53875