PREDICTING HOUSE PRICES USING MACHINE LEARNING

By
Abish
Amaresh
Gopi Krishna
Altrin
Ahilesh

Feature Selection

- This notebook tries various feature selection techniques.
- Based in these techniques, select features that contribute most to the output variable.
- Techniques used are
  - Correlation and Mutual Information for Numerical Features
  - SelectFromModel
  - SelectKBest

```
# Import starting libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib
import matplotlib.pyplot as plt

# Read the dataset and create original copies
X_train = pd.read_csv("../input/home-data-for-ml-course/train.csv")
X_test = pd.read_csv("../input/home-data-for-ml-course/test.csv")

X_train_original = X_train.copy()
X_test_original = X_test.copy()

pd.set_option("display.max_columns", None)
X_train.head()
```

```python
# Separate temporal features
feature_with_year = []

for feature in X_train.columns:
    if "Yr" in feature or "Year" in feature:
        feature_with_year.append(feature)

feature_with_year

# Separate numerical and categorial features
categorical_features = []
numerical_features = []
discrete_features = []
continuous_features = []

for feature in X_train.columns:
    if X_train[feature].dtypes == "O":
        categorical_features.append(feature)
    else:
        numerical_features.append(feature)
```

```
if len(X_train[feature].unique()) <= 20 and feature not in feature_with_year:
    discrete_features.append(feature)
else:
    continuous_features.append(feature)

print("Numerical Features ", numerical_features)
print("\n\nDiscrete Features ", discrete_features)
print("\n\nContinuous Features ", continuous_features)
print("\n\nCategorical Features ", categorical_features)

# Perform Feature Engineering (Already Done in previous notebook)
from sklearn.preprocessing import LabelEncoder, StandardScaler

numerical_features_with_null_values = []
categorical_features_with_null_values = []
feature_for_log_transform = ['LotFrontage', "LotArea", "1stFlrSF", "GrLivArea"]
```

```
def feature_engineering(X):
for feature in X.columns:
if X[feature].isna().sum() > 0 and feature in numerical_features:
numerical_features_with_null_values.append(feature)

if X[feature].isna().sum() > 0 and feature in categorical_features:
categorical_features_with_null_values.append(feature)

# Numerical Features
for feature in numerical_features_with_null_values:
if feature != "SalePrice":
X[feature].fillna(X[feature].median(), inplace=True)
# We choose median because of outliers

# Categorical Features
for feature in categorical_features_with_null_values:
X[feature].fillna("Others", inplace=True)

for feature in feature_with_year:
if feature != "YrSold":
X[feature] = X["YrSold"] - X[feature]
```

```
for feature in X.columns:
    if X[feature].dtypes != "O":
        q1 = np.percentile(X[feature], 25, interpolation='midpoint')
        median = np.percentile(X[feature], 50, interpolation='midpoint')
        q3 = np.percentile(X[feature], 75, interpolation='midpoint')
        iqr = q3 - q1

        upper_limit = (q3 + 1.5*iqr)
        lower_limit = (q1 - 1.5*iqr)

        if upper_limit != 0 and lower_limit != 0:
            X[feature] = np.where(X[feature] > upper_limit, median, X[feature])
            X[feature] = np.where(X[feature] < lower_limit, median, X[feature])

for feature in categorical_features:
    feature_percentage = X.groupby(feature)["SalePrice"].count()*100/len(X)
    feature_less_than_1 = feature_percentage[feature_percentage<1].index
    X[feature] = np.where(X[feature].isin(feature_less_than_1), "Others_R", X[feature])
```

```
encoder = LabelEncoder()
for feature in categorical_features:
X[feature] = encoder.fit_transform(X[feature])

# Convert each feature into float64 type
X = X.astype("float64")

for feature in feature_for_log_transform:
X[feature] = np.log1p(X[feature])


scaler = StandardScaler()
X[continuous_features] = scaler.fit_transform(X[continuous_features])


return X
X_train.shape
```

Output :

```
['YearBuilt', 'YearRemodAdd', 'GarageYrBlt', 'YrSold']

(1460, 81)
```