Chapter 3 – Feature Engineering

Table of Contents

Chapter 3 – Feature Engineering 1

Introduction to feature Engineering 1

Feature Selection 2

- Removal of Unused Columns 2
- Dropping the columns with Missing values 2
- Low variance features 2
- Multi collinearity 2
- p-value 3

Exercise 3.01: Feature Engineering Strategies 3

- Open the following link to get started (Empty Jupyter Environment) or start with your own local or Hosted Jupyter environment: 3
- RESOURCES AND REFERENCES 3
  - The code is located here: 3
  - The code repository is located here: 4

Feature Importance 4

Exercise 3.02: Feature Importance 5

- Open the following link to get started (Empty Jupyter Environment) or start with your own local or Hosted Jupyter environment 5
- RESOURCES AND REFERENCES 6
  - The code is located here: 6
  - The code repository is located here: 6
- Principal Component Analysis 6

Exercise 3.03: Principal Component Analysis 7

Automated Feature selection Techniques 7

- Chi-square based technique 7
- Regularization 7
- Sequential Selection 8

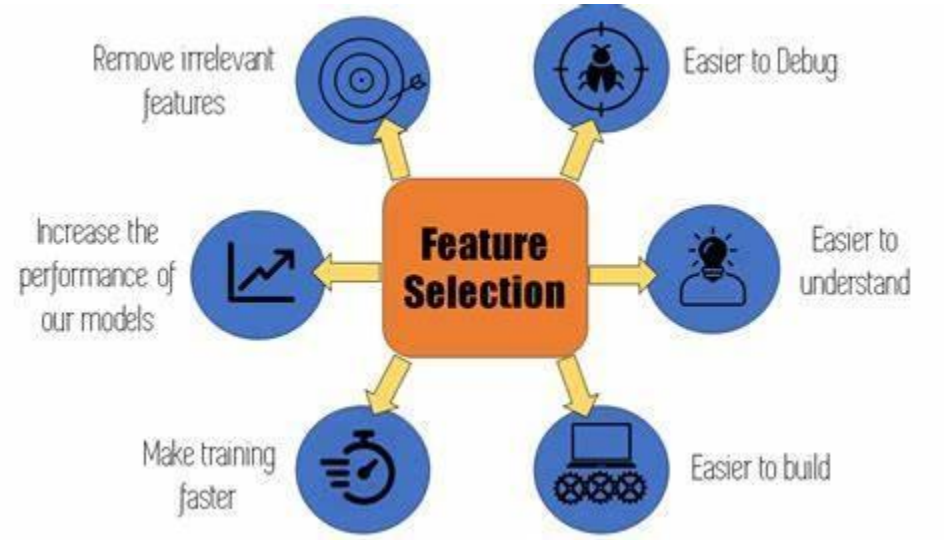
Exercise 3.04: Automated Feature selection 8
Open the following link to get started (Empty Jupyter Environment) or start with your own local or Hosted Jupyter environment: 8
RESOURCES AND REFERENCES 8
The code is located here: 8
The code repository is located here: 8
Binning 9
Bins Ranges 9
Exercise 3.05: Binning Features 9
Open the following link to get started (Empty Jupyter Environment) or start with your own local or Hosted Jupyter environment: 9
RESOURCES AND REFERENCES 9
The code is located here: 9
The code repository is located here: 9
Activity 3.01: Perform the Steps Mentioned Below: 10
Open the following link to get started (Empty Jupyter Environment) with the aforementioned instructions: 10
RESOURCES AND REFERENCES 10
In order to check the solution, you can access it from here: 10
Conclusion 10

Introduction to feature Engineering

The datasets have been explored in the last two chapters through EDA. They were analyzed briefly in terms of the usage of different types of analysis techniques. Now, in this chapter, we are going to cover another aspect of data processing. Engineering and its domains will be explored in this chapter. Dataset will be analyzed. Then after doing some ore processing on that data, we will be looking into the extraction of the features from that dataset. The main aim of feature engineering is to provide more information for the analysis being performed on the dataset.

It is observed that some unnecessary features decrease the training speed, and may decrease the generalization performance on the dataset. Whereas, this might be the case when some additional features are required.

Feature Engineering is useful in the terms as depicted here:



In this chapter, we will be looking into the dataset of “Miami Housing Prices” having 17 columns and 13932 rows. We will be applying the strategies of feature selection to this dataset.

Feature engineering is the process of using the knowledge of feature selection to extract the features from raw data. It is a step to be performed after EDA on the dataset. Data scientists are concerned with the iteration process to reduce errors and improve the accuracy of their models. Once the definition of the feature set is ready for practical use, the next step in feature engineering is to manufacture features in production.

You can learn more about Feature Selection here:
https://www.youtube.com/watch?v=5bHpPQ6_OU4

Feature Selection

This technique deals with keeping some features and letting others go. But there is some logic to deciding which features should be cut off. In the upcoming sections, we will be looking into some strategies to perform it:

Removal of Unused Columns

In most cases, it is apparent that some columns are not used for further usage. They may not appear in data preprocessing or other relevant techniques. It is quite obvious to remove such columns so that a more simplified version of data can be obtained.

Dropping the columns with Missing values

Sometimes, it appears that a large number of null values create problems with data handling. People apply different strategies to clean up missing data. But the frequently occurring missing values, it is preferred to drop the entire column.

Low variance features

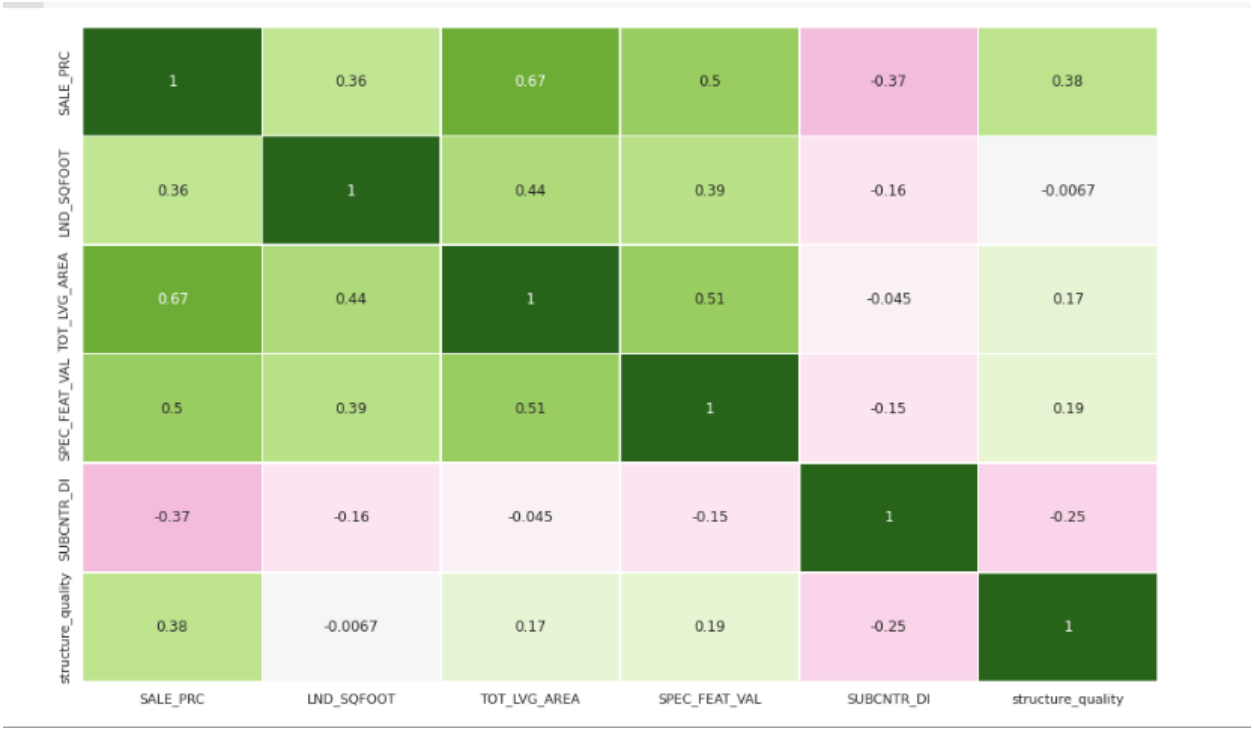
It is seen that a certain column may have values that are almost the same. And it would have 0 variances. Such a column can be dropped due to the low variance features. An ideal candidate can be the one having the lowest variance. The variances of each variable can be checked as:

```
# variance of numeric features
(df.select_dtypes(include=np.number).var().astype('str'))
```

Multi collinearity

When there is a correlation between any two features, multi-collinearity arises. Mostly it is expected that the two features are independent. Thus no collinearity exists between them. You will be seeing in the upcoming exercise about this scenario that "**TOT_LVG_AREA"** and "**LND_SQFOOT**" are more correlated with **SALE_PRC**. So you can eliminate one of them and let some other features predict the target variable.

It is shown as:



p-value

It is an indicator of the relationship between a predictor and a target. ***Statsmodels*** library gives us a summary of regression outputs which includes feature coefficients and the relevant p-values. The insignificant features can be removed one by one and the model is re-run each time until a set of features with significant p values and improved performance is achieved.

Note: You can practice the above-mentioned features through an upcoming exercise. Follow the steps mentioned in a guided exercise to practice these features

Exercise 3.01: Feature Engineering Strategies

In this exercise, we will be using the Miami House prices prediction–based dataset. We will be working on the features using the above-mentioned strategies to extract the features and work on them.

Open the following link to get started (Empty Jupyter Environment) or start with your own local or Hosted Jupyter environment:

https://mybinder.org/v2/gh/fenago/jupyter/HEAD

RESOURCES AND REFERENCES

The code is located here:

https://github.com/fenago/datawrangling/blob/main/Chapter%203/Exercise_3_01_Feature_Selection.ipynb

The code repository is located here:

https://github.com/fenago/datawrangling/tree/main/Chapter%203

Feature Importance

As we have already worked on some feature selection techniques. Feature importance is an essential domain of the feature selection technique that helps in understanding it better through the following ways:

- SelectKBest
- Linear regression
- Random Forest
- XGBoost
- Recursive Feature Elimination

Linear regression, SelectKBest, and Decision Tree classification will be practiced through the upcoming exercises.

A ***decision tree*** splits the data using a particular feature that contributes toward decreasing the impurity. Thus finding the best feature is an important part of how the algorithm works in a classification task. Moreover, an attribute, ***feature_importances_*** can be used to access the best features.

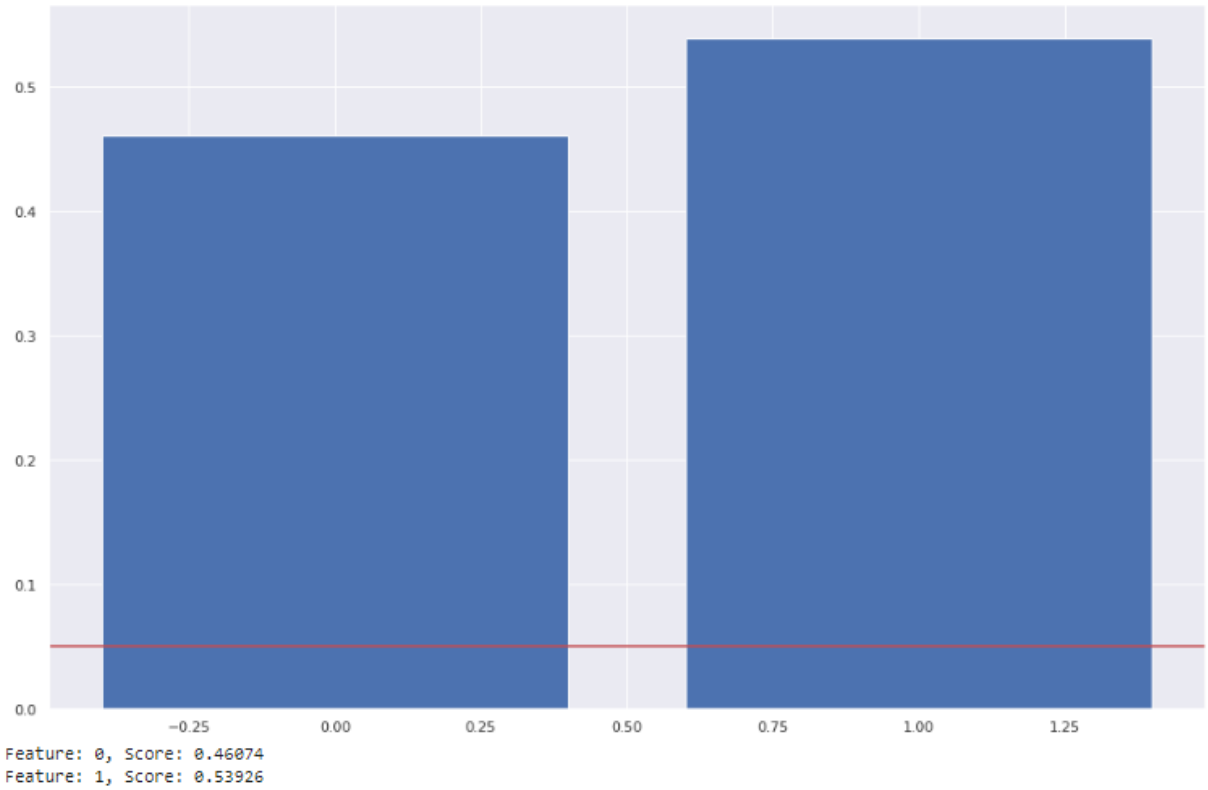
We have covered it in the following way:

```
from sklearn.tree import DecisionTreeClassifier #Decision Tree
model = DecisionTreeClassifier()
model.fit(X_train, y_train)
# get importance
importance = model.feature_importances_
# summarize feature importance
for i,v in enumerate(importance):
    print('Feature: %0d, Score: %.5f' % (i,v))
```

Plots were also built using the following snippet:

```
# plot feature importance
plt.bar([x for x in range(len(importance))], importance)
plt.axhline(y=0.05, color='r', linestyle='-')
plt.show()
#use only high important features to feed into a model
for i,v in enumerate(importance):
    if v >= 0.05:
        print('Feature: %0d, Score: %.5f' % (i,v))
```

The following output was obtained from the above code snippet:



Note: You can practice the above-mentioned methodologies through an upcoming exercise. Follow the steps mentioned in a guided exercise to practice these features

Exercise 3.02: Feature Importance

In this exercise, we will be using the Miami House prices prediction–based dataset. We will be working with the features using the feature_importances _ attribute in this exercise:

Open the following link to get started (Empty Jupyter Environment) or start with your own local or Hosted Jupyter environment

https://mybinder.org/v2/gh/fenago/jupyter/HEAD

RESOURCES AND REFERENCES

The code is located here:

https://github.com/fenago/datawrangling/blob/main/Chapter%203/Exercise_3_02_Feature_Importance.ipynb

The code repository is located here:

https://github.com/fenago/datawrangling/tree/main/Chapter%203

Principal Component Analysis

Principal Component Analysis (PCA) is a frequently used technique by data scientists to make a more efficient model training. It helps in visualizing the data in lower dimensions. Its main purpose is to reduce the dimensionality of high-dimensional feature space.

The original features are projected into new dimensions here. The main aim is to find the number of components that can better illustrate the variance of the data.

It is done through the following snippet:

```
# import PCA module
from sklearn.decomposition import PCA
# scaling data
X_scaled = scaler.fit_transform(X)
# fit PCA to data
pca = PCA()
pca.fit(X_scaled)
evr = pca.explained_variance_ratio_
# visualizing the variance explained by each principal component
s
plt.figure(figsize=(12, 5))
plt.plot(range(0, len(evr)), evr.cumsum(), marker="o", linestyle
="--")
plt.xlabel("Number of components")
plt.ylabel("Cumulative explained variance")
```

Note: You can practice the above-mentioned strategy through an upcoming exercise. Follow the steps mentioned in a guided exercise to practice these features

Exercise 3.03: Principal Component Analysis

In this exercise, we will be using the Miami House prices prediction–based dataset. We will be working with the features using the above-mentioned strategies to extract the features and work on them.

Open the following link to get started (Empty Jupyter Environment) or start with your own local or Hosted Jupyter environment

https://mybinder.org/v2/gh/fenago/jupyter/HEAD

RESOURCES AND REFERENCES

The code is located here:

https://github.com/fenago/datawrangling/blob/main/Chapter%203/Exercise_3_03_PCA.ipynb

The code repository is located here:

https://github.com/fenago/datawrangling/tree/main/Chapter%203

Automated Feature Selection Techniques

Sklearn library of python has built-in methodologies to cover feature selection. It provides an entire module to deal with the feature selection. Some such automated processes within ***sklearn*** are:

Chi-square based technique

The chi-squared-based technique selects a certain number of user-defined features based on the total number of records and their absolute values. These scores are determined by computing chi-squared statistics between X (independent) and y (dependent) variables.

You can use *sklearn* to determine the number of features you want to keep. If you want to keep all features, implementation will look like this:

```
# select K best features
X_best = SelectKBest(chi2, k='all').fit_transform(X,y)
# number of best features
X_best.shape[1]
```

Regularization

It is used to reduce overfitting. If you are having a lot of features, regularization controls their effect either by setting feature coefficients to zero – ***L1 Regularization*** or by shrinking feature coefficients – ***L2 Regularization***.

We have implemented a ***LinearSVC*** algorithm with ***penalty = 'L1".*** Implementation is as follows:

```
# implement algorithm
from sklearn.svm import LinearSVC
model = LinearSVC(penalty= 'l1', C = 0.002, dual=False)
model.fit(X,y)
# select features using the meta transformer
selector = SelectFromModel(estimator = model, prefit=True)
```

```
X_new = selector.transform(X)
X_new.shape[1]

# names of selected features
feature_names = np.array(X.columns)
feature_names[selector.get_support()]
```

Sequential Selection

This is one of the classical statistical techniques. One feature can be added or removed at a time and model performance is checked until it is optimized enough to meet your needs. This technique has two variants. The forward selection technique starts with a 0 feature. One feature is then added which minimizes the error, then another feature is added, and so on.

On the other hand, the backward selection is the opposite. The model starts with all features and calculates the error. Then one feature is eliminated and the process continues until the desired number of features remains.

Note: You can practice the above-mentioned features through an upcoming exercise. Follow the steps mentioned in a guided exercise to practice these features

Exercise 3.04: Automated Feature selection

In this exercise, we will be using the Miami House prices prediction–based dataset. We will be working with the features using the above-mentioned strategies to practice the automated feature selection techniques.

Open the following link to get started (Empty Jupyter Environment) or start with your own local or Hosted Jupyter environment:

https://mybinder.org/v2/gh/fenago/jupyter/HEAD

RESOURCES AND REFERENCES

The code is located here:

https://github.com/fenago/datawrangling/blob/main/Chapter%203/Exercise_3_04_Automated_Feature_Selection_Techniques.ipynb

The code repository is located here:

https://github.com/fenago/datawrangling/tree/main/Chapter%203

Binning

Binning is a method that takes a column with continuous numbers and places the numbers in "**bins**" based on pre-defined ranges. In this way, a new categorical variable feature is achieved. Let us say, we are using the column, **SALE_PRC** from our dataset. It is indicating the prices of houses, and we will be adding some random values of our own to adjust them into the bins. And we will make 3 bins for it as: "`[200000, 400000, 390000]`".

Bins Ranges

Firstly, the ranges for these bins need to be determined. There are many ways to do so. One way is to divide the bins up evenly on the basis of the distribution of values. It can be visualized in terms of histograms by passing one parameter of bins to **plt.hist()** method.

Note: You can practice the above-mentioned features through an upcoming exercise. Follow the steps mentioned in a guided exercise to practice these features

Exercise 3.05: Binning Features

In this exercise, we will be using the Miami House prices prediction–based dataset. We will be working on the features using the above-mentioned strategies to put certain values of a feature into a bin with ranges.

Open the following link to get started (Empty Jupyter Environment) or start with your own local or Hosted Jupyter environment:

https://mybinder.org/v2/gh/fenago/jupyter/HEAD

RESOURCES AND REFERENCES

The code is located here:

https://github.com/fenago/datawrangling/blob/main/Chapter%203/Exercise_3_05_Binning.ipynb

The code repository is located here:

https://github.com/fenago/datawrangling/tree/main/Chapter%203

Activity 3.01: Perform the Steps Mentioned Below:

For this activity, you have to use a Real estate Pricing dataset, and work on its features. Binning will be done here for the Prices. Follow the Steps mentioned below to perform the task:

1. Open the colab notebook and import the necessary libraries
2. Load the given dataset.
3. Then look for the shape of the dataset and print values using the head function.
4. Use nunique() to find out the number of unique values over the column axis of "Y house price of unit area".
5. Use value_counts() function to count the total values of the same feature.
6. Then plot the count plots.
7. Then perform binning, by making another column for the same feature.
8. Create a copy of the feature in that new column
9. Then add some new values.

Note: Firstly perform the above-mentioned Steps, then open up the solution to consult your way of performance.

Open the following link to get started (Empty Jupyter Environment) with the aforementioned instructions:
https://mybinder.org/v2/gh/fenago/jupyter/HEAD

RESOURCES AND REFERENCES

In order to check the solution, you can access it from here:
https://github.com/fenago/datawrangling/blob/main/Chapter%203/Activity_3_01_Solution.ipynb

Conclusion

In this chapter, we have covered all the main techniques and strategies used for feature engineering. Core concepts with exercises have been covered. It was seen that feature engineering techniques require features to be used in an efficient manner. Moreover, they must be analyzed briefly before their extraction.

Binning variables were also seen in detail. These techniques can help data scientists to organize their datasets on the basis of certain features so that the results of post-processing techniques must be optimized. Such techniques are applied prior to fitting the model such as dropping columns with missing values, columns with multi-collinearity, and dimensionality reduction with

PCA. While some other techniques must be pursued after base model implementation such as feature coefficients, p-value, and VIF.

In the upcoming chapters, we will be looking into the Encoding and normalizing techniques. We will learn how different features can be encoded.