Predictive Analytics

ALY 6020, CRN 80405

Professor Vladimir Shapiro

Predictive Analytics - Midterm Test

Submitted By - Richa Umesh Rambhia

Midterm Exam - Predict Price of a Candy Bar

Table of Contents

Theory Questions

For all questions 1-4, please provide the answer and 2-3 sentences around the rationale of why that is the answer.

Q1. Name three acceptable methods for replacing a missing continuous value when less than 20% of the data is missing. Provide the reasoning of why that works and when you would use this technique.

Q2. Rank the statistical values in order of importance (most to least) when evaluating a linear regression model (R2, p-Value, coefficients).

Q3. In what situation would you use R2 vs. MSE as a measure to understand the fit of a multiple linear regression model? Provide a 2-3 sentence explanation.

Q4. What are three things we can learn from a correlation plot, and give an example of how each reason you've listed is important in the model-building process.

Answering Question 1 to Question 4:

Q1. Name three acceptable methods for replacing a missing continuous value when less than 20% of the data is missing. Provide the reasoning of why that works and when you would use this technique.

A1. The three acceptable methods for replacing a missing continuous value when less than 20% of the data is missing are as follows:

1. **Mean/Median Imputation**
2. **Regression Imputation**
3. **Pattern Substitution**

Mean/Median Imputation:

- The mean/median imputation method involves replacing missing values with either of the imputations of the non-missing data points, which means the missing values are replaced by either the mean or the median of the non-missing data values present in the dataset.
- This method is used when less than 20% of the data is missing and when the data is completely missing at random.
- The selection of whether to choose a mean imputation or a median imputation depends on the distribution of data, i.e., when the data points are numerical and normally distributed, mean imputation method can be used to replace the missing values, whereas when the distribution contains outliers or it is skewed, median imputation is used because median will be less sensitive to the outliers in the data than the mean.

Regression Imputation:

- The regression imputation method is used when the data is missing at random and involves using a regression model itself to predict and replace the missing values.
- This method can be used when there is a strong correlation between the other variables of the dataset and the data points that are missing, as this will accurately predict and replace the non missing values based on the regression model built.

Pattern Substitution:

- This method refers to replacing the missing data values with a known pattern of values that are present in the dataset and match the missing values.
- This is a simple way to replace the missing values based on analyzing the patterns of data points of the non-missing values.
- Pattern Substitution is used when the data points are not missing at random and when there is a pattern that can be determined to replace the missing values.

Q2. Rank the statistical values in order of importance (most to least) when evaluating a linear regression model (R2, p-Value, coefficients).

A2. The statistical values in order of importance (most to least) when evaluating a linear regression model are as follows.

1. **R-squared (R2):** R-squared or R2 is considered the most important statistical value when evaluating a linear regression model or goodness of fit of the model. It is a measure that helps to determine of how the linear regression model fits the data, where a high R-squared value indicates a good fit of the model, and hence it is an important value to be considered when evaluating the model as it determines the quality of fit of the data on the model.
2. **p-value:** p-value in a linear regression model is the probability of determining the relationship between the independent and dependent variables. p-value less than 0.05 which is the significance level indicates that the variable is statistically significant, whereas if the p-value is greater than 0.05 it means that the variable is statistically insignificant and should not be considered for analysis. This helps in understanding the statistical relationship between the variables and thus is important while evaluating a regression model to analyze the variables which are significant for prediction.
3. **Coefficients:** Coefficients in the linear regression model are used to determine the effect of the independent variables on the target variable. This value help in understanding both the magnitude and direction of a variable which is used to determine the relationship between the independent and dependent variables. For example, if a positive coefficient is obtained for a variable, it indicates that the variable has a positive influence on the target variable, whereas if the coefficient value is negative, it means that the variable has a negative influence on the dependent variable. Hence, coefficients are important in evaluating the linear regression model as they help in analyzing the features that are contributing in the prediction of the target variable.
4. **Standard Error:** Standard error measures the precision of the coefficients indicating how much the coefficient value is likely to vary from the true coefficient when the model is fitted to a different set of data. This is the least important statistical value considered while evaluating the regression model.

Q3. In what situation would you use R2 vs. MSE as a measure to understand the fit of a multiple linear regression model? Provide a 2-3 sentence explanation.

A3.

- **R-squared** is the measure of proportion of variance in the dependent variable that is explained by the independent variables in the model building. This measure helps in understanding how well the model fits the data, where a high R2 value indicates a good fit model and a low R2 value indicates a poor fit model.
- **MSE, (Mean Squared Error)** is the average squared distance between the predicted values of the model and the actual values of the dataset. This metric is useful to measure the efficiency of the prediction on new values of the dataset.
- Therefore, in situation when the aim is to **evaluate the goodness of fit of the model, R-squared value** is used, whereas when we want to **evaluate the accuracy and efficiency of the predictions made on the test data by the model, MSE score** is used to understand the fit of a multiple linear regression model.

Q4. What are three things we can learn from a correlation plot, and give an example of how each reason you've listed is important in the model-building process.

A4. Correlation Plot is a visual representation of the variables present in the dataset which helps in understanding the relationship between the different variables and how highly the variables are corelated to each other. The values of the correlation plot range from **-1 to 1**, where -1 indicates a **negative correlation** between the variables, 0 indicates **no correlation**, and 1 indicates a **positive correlation**.

The three things that we can learn from a correlation plot which are important in the model building process are as follows:

1. Correlation plot helps to analyze the **relationship strength** between each of the variables of the dataset. This relationship between the variables is determined by the correlation value and this is important in the model building process as the variables that are highly correlated to each other can be excluded from the training. Hence, the first thing that we learn from the correlation plot is that it helps us to analyze the correlation between each of the variables in the dataset and this is important to avoid multicollinearity and improve model efficiency.

2. The second thing that we learn from the correlation plot is the **direction of the relationship between the variables**, that can be positive or negative, which means that it helps in analyzing whether a variable is postivitely correlated to another variable, negatively correlated to another variable, or has a neutral relation with the variables. This is important in the model building process as it will help to model and fit the relationship in the model training such that if a negative correlation is present between the variables, the negative coefficients can be included in the model for training as negative coefficients also influence the target variable.
3. Lastly, the correlation plot tells us about the **form of the relationship or outliers present in the dataset**, which means that the plot helps in determining the potential outliers that may impact the predicitions and the overall performance of the model. This is important in the model building process as it may reduce the accuracy and efficiency of the model, and hence the outliers need to be analyzed in order to decide whether or not to remove them based on their impact on the model and predictions.

Implementation of Linear Regression model

Introduction

Linear Regression Model

Machine Learning algorithms are classified into *supervised and unsupervised* learning, where supervised learning algorithms are further classified into **Classification & Regression** based problems. Classification problems deal with *categorical data* in order to **classify the classes** for the data points, whereas Regression problems are **prediction** based models that are *continuous* in nature and predict the output variable depending on the features of the data. [4]

Simple Linear Regression Model Linear regression models are simple methods that are used for the predictive analysis that show the linear relationship between the independent variable of the dataset and the target variable.

Multiple Linear Regression Model Simple linear regression models are used when there is only 1 independent variable to predict the target variable. However, multiple linear regression model is used when there are multiple independent variables in order to predict a single dependent variable. [4]

Problem Statement

Questions 5-8 are based on the dataset attached to the exam description. Answers require coding. Your goal is to predict pricepercent, and we are trying to see if we can predict the price of a candy bar based on its features.

Predicting the pricepercent of a candy bar

The project deals with the prediction of the price percent for a candy bar based on the various attributes that are present in the dataset such that the higher the value, the more expensive the candy bar is. The aim here is to understand the parameters and features that are affecting the price of a candy bar such that recommendations can be given to the company based on the features that affect the price range and the company can price it in the range as per the recommended features.

Analysis

Your goal is to predict pricepercent, and we are trying to see if we can predict the price of a candy bar based on its features.

Installing required packages

In [1]:
```
!pip install pandas_profiling
!pip install featurewiz
```

```
Looking in indexes: https://pypi.org/simple, (https://pypi.org/simple,) https://us-python.pkg.dev/colab-wheels/public/simple/ (https://us-python.pkg.dev/colab-wheels/public/simple/)
Collecting pandas_profiling
  Downloading pandas_profiling-3.6.6-py2.py3-none-any.whl (324 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 324.4/324.4 kB 6.2 MB/s eta 0:00:00
Collecting ydata-profiling
  Downloading ydata_profiling-4.1.2-py2.py3-none-any.whl (345 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 345.9/345.9 kB 30.7 MB/s eta 0:00:00
Collecting multimethod<1.10,>=1.4
  Downloading multimethod-1.9.1-py3-none-any.whl (10 kB)
Requirement already satisfied: seaborn<0.13,>=0.10.1 in /usr/local/lib/python3.10/dist-packages (from ydata-profiling->pandas_profiling) (0.12.2)
Collecting tqdm<4.65,>=4.48.2
  Downloading tqdm-4.64.1-py2.py3-none-any.whl (78 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 78.5/78.5 kB 8.0 MB/s eta 0:00:00
Requirement already satisfied: numpy<1.24,>=1.16.0 in /usr/local/lib/python3.10/dist-packages (from ydata-profiling->pandas_profiling) (1.22.4)
Collecting phik<0.13,>=0.11.1
  Downloading phik-0.12.3-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (679 kB)
```

Importing libraries

In [2]:
```
import pandas as pd
import numpy as np
import pandas_profiling
import ydata_profiling
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.impute import KNNImputer
from featurewiz import featurewiz
from sklearn.preprocessing import LabelEncoder
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression, Lasso
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
```

```
Imported version = 0.1.55.
from featurewiz import FeatureWiz
wiz = FeatureWiz(verbose=1)
X_train_selected = wiz.fit_transform(X_train, y_train)
X_test_selected = wiz.transform(X_test)
wiz.features  ### provides a list of selected features ###
```

Loading the dataset

In [3]:
```
chocolate_data = pd.read_csv("Chocolate.csv")
chocolate_data
```

Out[3]:

	competitorname	chocolate	fruity	caramel	peanutyalmondy	nougat	crispedricewafer	hard	bar	Multiple Pieces	sugarpercent	priceperc
0	100 Grand	1	0	1.0	0	0	1	0	1	0.0	0.732	0.8
1	3 Musketeers	1	0	0.0	0	1	0	0	1	0.0	0.604	0.
2	One dime	0	0	0.0	0	0	0	0	0	0.0	0.011	0.
3	One quarter	0	0	0.0	0	0	0	0	0	0.0	0.011	0.
4	Air Heads	0	1	0.0	0	0	0	0	0	0.0	0.906	0.
...	...	...	...	...	...	...	...	...	...	...	...	
80	Twizzlers	0	1	0.0	0	0	0	0	0	0.0	0.220	0.
81	Warheads	0	1	0.0	0	0	0	1	0	0.0	0.093	0.
82	WelchÕs Fruit Snacks	0	1	0.0	0	0	0	0	0	1.0	0.313	0.
83	WertherÕs Original Caramel	0	0	1.0	0	0	0	1	0	0.0	0.186	0.
84	Whoppers	1	0	0.0	0	0	1	0	0	1.0	0.872	0.8

85 rows × 12 columns

Table 1. Chocolate Price Prediction Data

Step 1: Exploratory Data Analysis

EDA is performed on the data in order to analyze various parameters and features of the dataset and to understand the *structure* of the dataset such that various *trends and patterns* between the variables is known. Exploratory Data Analysis helps in understanding the *relationship between the various independent and dependent variables* of the dataset that would further be useful in building the model such as description analysis and statistical analysis.

Descriptive Analysis

In [ ]:
```
# displaying number of rows and columns
print("Total number of Rows and Columns:", chocolate_data.shape)

print("\n-----------------------------------------------------------")

# displaying field values/column names
print("\nColumn Names:\n")
chocolate_data.columns
```

```
Total number of Rows and Columns: (85, 12)

-----------------------------------------------------------

Column Names:
```

Out[171]:
```
Index(['competitorname', 'chocolate', 'fruity', 'caramel', 'peanutyalmondy',
       'nougat', 'crispedricewafer', 'hard', 'bar', 'Multiple Pieces',
       'sugarpercent', 'pricepercent'],
      dtype='object')
```

In [ ]:
```
# displaying data types
print("Data types:\n")
chocolate_data.dtypes
```

```
Data types:
```

Out[172]:
```
competitorname       object
chocolate             int64
fruity                int64
caramel             float64
peanutyalmondy        int64
nougat                int64
crispedricewafer      int64
hard                  int64
bar                   int64
Multiple Pieces     float64
sugarpercent        float64
pricepercent        float64
dtype: object
```

From the *descriptive analysis*, it is observed that there are total **85 rows of data** and **12 field values** and the data type for each of the field value is displayed in order to understand what data type values are present in the dataset.

Here, as it is observed there are different types of data points that are present in the dataset which are **numerical data type** having *'int' and 'float'* values and remaining field values are of **object type**, which needs to be updated to **category** type as per the values present in the dataset, later in the preprocessing stage.

Statistical Analysis

In [ ]:
```
# dataset info
print("Dataset Info:\n")
chocolate_data.info()
```

```
Dataset Info:

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 85 entries, 0 to 84
Data columns (total 12 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   competitorname    85 non-null     object 
 1   chocolate         85 non-null     int64  
 2   fruity            85 non-null     int64  
 3   caramel           81 non-null     float64
 4   peanutyalmondy    85 non-null     int64  
 5   nougat            85 non-null     int64  
 6   crispedricewafer  85 non-null     int64  
 7   hard              85 non-null     int64  
 8   bar               85 non-null     int64  
 9   Multiple Pieces   61 non-null     float64
 10  sugarpercent      85 non-null     float64
 11  pricepercent      82 non-null     float64
dtypes: float64(4), int64(7), object(1)
memory usage: 8.1+ KB
```

Table 2. Information about the dataset

In [ ]:
```
# describing the dataset
print("Describing the dataset:\n")
round(chocolate_data.describe(),1)
```

```
Describing the dataset:
```

Out[174]:

	chocolate	fruity	caramel	peanutyalmondy	nougat	crispedricewafer	hard	bar	Multiple Pieces	sugarpercent	pricepercent
count	85.0	85.0	81.0	85.0	85.0	85.0	85.0	85.0	61.0	85.0	82.0
mean	0.4	0.4	0.2	0.2	0.1	0.1	0.2	0.2	0.5	0.5	0.5
std	0.5	0.5	0.4	0.4	0.3	0.3	0.4	0.4	0.5	0.4	0.3
min	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
25%	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.3	0.3
50%	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	0.5
75%	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.7	0.7
max	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	3.0	1.0

Table 3. Dataset Description

Statistical Analysis helps in understanding about each of the numerical field type based on the **total count values, minimum value, maximum value, standard deviation**, etc. giving an overall analysis of the field data points about the various rows present in the dataset.

For example, as observed in the chocolate dataset, we see that there are multiple field values having the *minimum, maximum values* along with the *total count of values* which is **85** and *standard deviation* of the column values. It can be observed that the maximum value of *Pricepercent* is **1.0** and the maximum value of *Sugarpercent* is **3.0**.

Thus, similarly, other parameters of the dataset can be analyzed based on their statistical values.

Data Profiling

In [4]:
```
chocolate_data_report = chocolate_data.profile_report(title='Chocolate Data Analysis Report', explorative = T
chocolate_data_report
```

```
Summarize dataset:   0%|          | 0/5 [00:00<?, ?it/s]

Generate report structure:   0%|          | 0/1 [00:00<?, ?it/s]

Render HTML:   0%|          | 0/1 [00:00<?, ?it/s]
```

Out[4]:

In [5]:
```
# Saving the profile report
chocolate_data_report.to_file(output_file="Chocolate Data Analysis Report.html")
```

```
Export report to file:   0%|          | 0/1 [00:00<?, ?it/s]
```

The data profiling report generated for the dataset helps in understanding various parameters such as the data type of the field values, the missing and duplicate values present in the dataset, the correlation between each of the field value, and the analysis of each of the field value on a individual basis based on correlation plot, histogram, and interaction graphs.

From the profiling report, it is observed that there are **2 numerical variable type and 10 categorical data type** field values present in the dataset of which the numerical data type have **integer and float values**. In this dataset, there are **missing values** present in the dataset, and the missing values visualization or plot also helps in understanding that there are some missing values present in the field values of the dataset. The percentage of duplicate rows is 0% indicating that there are no duplicate values present in the dataset and for each field value a separate visualization is displayed in order to specifially analyze a particular field value.

Further cleaning of the data is implemented in the below steps.

Step 2: Data Cleaning

Q5. Perform data cleansing. Which columns have missing or unclean data? How would you handle cleaning those up? Are there any outliers? Discuss in 4-5 sentences and have the code.

---

1. Checking for null values in each column of the dataset, i.e., missing values
2. Replacing missing values using various imputation methods
3. Checking for incorrect data types in field values and correcting the data type of the column
4. Renaming the column names
5. Checking for outliers in the dataset
   a. *Boxplot*
   b. *Distribution Plot*

---

1. Checking for null values in each column of the dataset, i.e., missing values

In [ ]:
```
for x in range(12):
    print("%-45s %10d" % (chocolate_data.columns.values[x], chocolate_data.iloc[:,x].isna().sum()))
```

```
competitorname                                0
chocolate                                     0
fruity                                        0
caramel                                       4
peanutyalmondy                                0
nougat                                        0
crispedricewafer                              0
hard                                          0
bar                                           0
Multiple Pieces                              24
sugarpercent                                  0
pricepercent                                  3
```

Table 4. Missing or null values

A5.

As it can be observed, there are missing values or null values present in the field value of the dataset, i.e., **'Caramel', 'Multiple Pieces', and 'Pricepercent'.** The missing values are analyzed further based on the data points and the distribution of data in order to implement the imputation methods for missing values.

Thus, the columns that have missing or unclean data are as follows:

1. Caramel
2. Multiple Pieces
3. Pricepercent

2. Replacing missing values using various imputation methods

In [ ]:
```
# displaying unique data

print("Displaying the unique data present in columns\n")
chocolate_data.nunique()
```

```
Displaying the unique data present in columns
```

Out[176]:
```
competitorname      85
chocolate            2
fruity               2
caramel              2
peanutyalmondy       2
nougat               2
crispedricewafer     2
hard                 2
bar                  2
Multiple Pieces      2
sugarpercent        36
pricepercent        29
dtype: int64
```

Table 5. Unique Data Count

In [ ]:
```
# checking for data distribution of field values 'Caramel', 'Multiple pieces', and 'Pricepercent'


# a. distribution plot for Caramel & Multiple pieces

plt.figure(figsize=(16,5))
plt.subplot(1,2,1)
sns.distplot(chocolate_data['caramel'])
plt.subplot(1,2,2)
sns.distplot(chocolate_data['Multiple Pieces'])
plt.show()
```

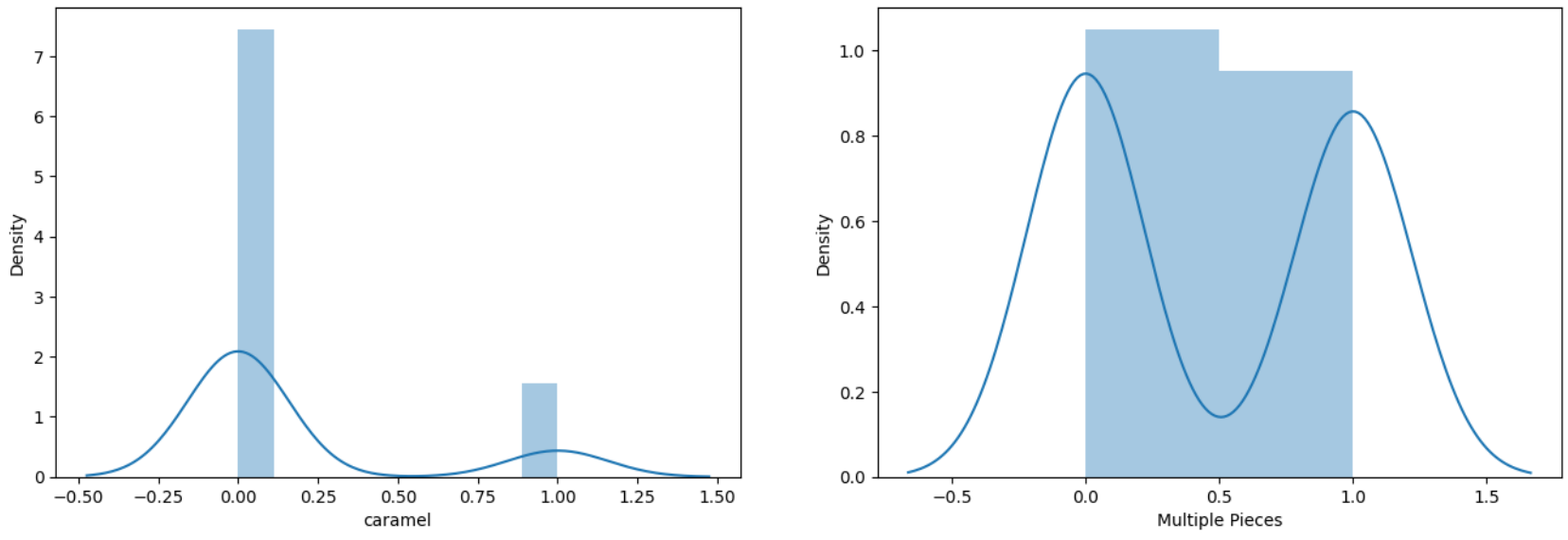

Figure 1. Distribution Plot for Caramel & Multiple pieces

In [ ]:
```
# checking for data distribution of field values 'Caramel', 'Multiple pieces', and 'Pricepercent'


# b. distribution plot for Pricepercent

plt.figure(figsize=(16,5))
plt.subplot(1,2,1)
sns.distplot(chocolate_data['pricepercent'])
plt.show()
```

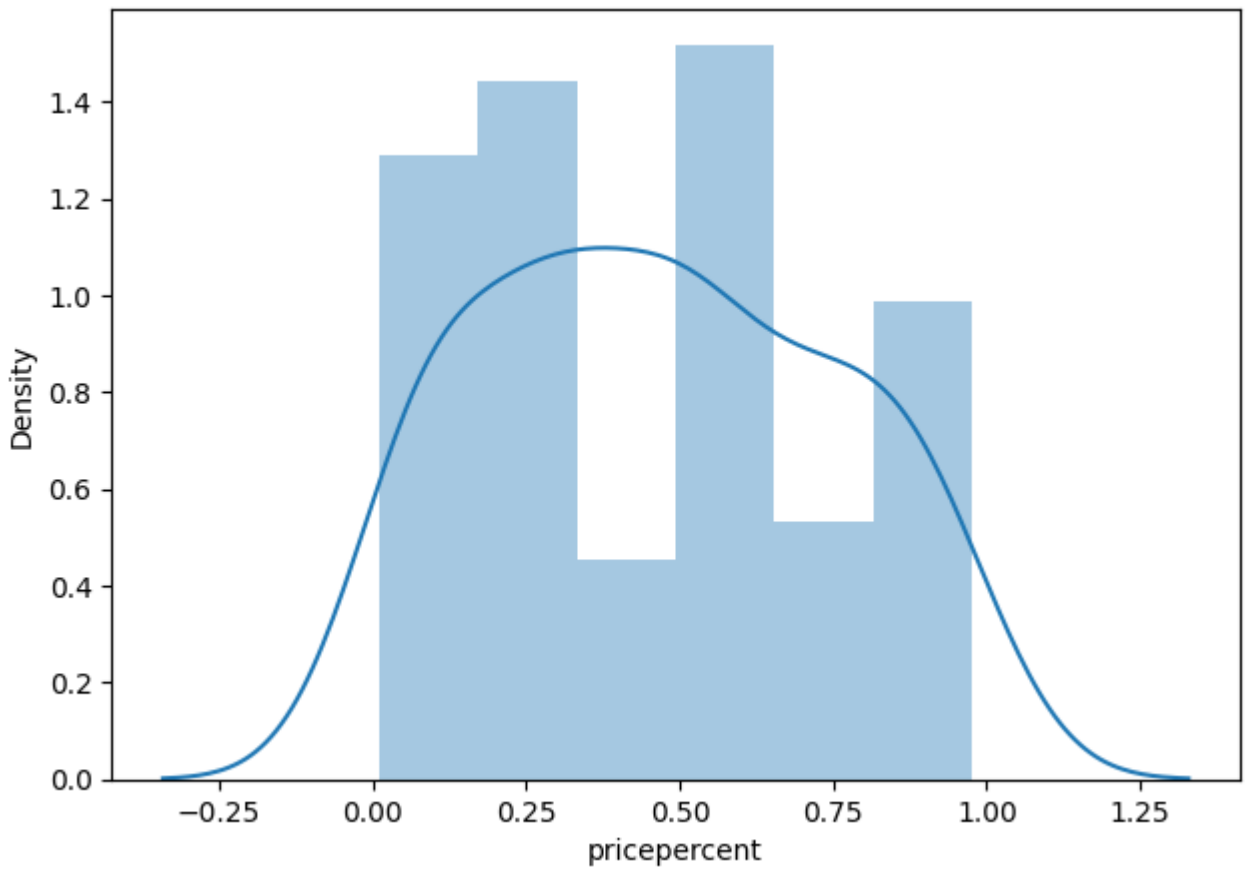

Figure 2. Distribution Plot for Pricepercent

The above distribution plots indicate that the data points for all the three variables are normally distributed, hence to perform a mean/median/mode imputation, the mean imputation method can be implemented as the data is normally distributed around the mean values.

A5.

The total number of rows of data present in the dataset are only 85 data points, and hence dropping a value for which the data is missing will not be efficient as there will less of data available for training of the model. Hence, none of the data values or rows of data are dropped from the dataset.

To handle cleaning and replacing the missing values present in the field values of the dataset, following steps are implemented.

a. KNN Imputation

b. Replacing missing values with 0

---

a. KNN Imputation for replacing missing values in Pricepercent column

In [4]:
```
# KNN imputation method for replacing missing values in Pricepercent column

imputer = KNNImputer(n_neighbors=3)
chocolate_data['pricepercent'] = imputer.fit_transform(chocolate_data[['pricepercent']])
print("Imputation successful.")
```

```
Imputation successful.
```

Since the '**Pricepercent**' column has missing values, and also is the target variable, using a mean or median imputation would not be efficient to use, and the data points are missing at random. Hence, in order to *efficiently impute the missing values*, **KNN Imputation** method is used as shown in the code above to replace the missing values of the 'Pricepercent' column.

b. Replacing missing values in 'Caramel' and 'Multiple pieces' column with 0 values

In [5]:
```
chocolate_data = chocolate_data.fillna(0)
print("Missing values replaced.")
```

```
Missing values replaced.
```

Here, there are missing values present in the 'Caramel' and 'Multiple pieces' column. These columns indicate whether the candy bar have caramel present or not and whether the bar has multiple pieces or not. This means that the data points in these columns are either 0 or 1. Hence, imputing it with mean or median value will not be appropriate based on the data values present.

Secondly, the missing values in these columns may indicate that the data is not present for these data points or that the particular feature is not present in the candy bar. Thus, the missing values or null values in **'Caramel' and 'Multiple pieces' column are replaced with 0 values.**

In [ ]:
```
# checking for any missing values after data cleaning & imputations

for x in range(12):
    print("%-45s %10d" % (chocolate_data.columns.values[x], chocolate_data.iloc[:,x].isna().sum()))
```

```
competitorname                                0
chocolate                                     0
fruity                                        0
caramel                                       0
peanutyalmondy                                0
nougat                                        0
crispedricewafer                              0
hard                                          0
bar                                           0
Multiple Pieces                               0
sugarpercent                                  0
pricepercent                                  0
```

Table 6. Missing value count after data cleaning

Thus, the data is cleaned and all the missing or null values are addressed and replaced as per the requirements of data points using suitable imputation methods, as observed in the code above.

3. Renaming the column names

In [6]:
```
column_names = {i: i.capitalize() for i in chocolate_data.columns}
chocolate_data = chocolate_data.rename(columns=column_names)

print("Renaming successful.")
```

```
Renaming successful.
```

In [ ]:
```
# column names after renaming

chocolate_data.columns
```

Out[204]:
```
Index(['Competitorname', 'Chocolate', 'Fruity', 'Caramel', 'Peanutyalmondy',
       'Nougat', 'Crispedricewafer', 'Hard', 'Bar', 'Multiple pieces',
       'Sugarpercent', 'Pricepercent'],
      dtype='object')
```

4. Checking for incorrect data types in field values and correcting the data type of the column

In [7]:
```
# correcting the data types for the variables of the dataset which are of object type to string/category type

chocolate_data['Competitorname'] = chocolate_data['Competitorname'].astype('category')
chocolate_data['Caramel'] = chocolate_data['Caramel'].astype('int')
chocolate_data['Multiple pieces'] = chocolate_data['Multiple pieces'].astype('int')
print("Data Type conversion successful.")
```

```
Data Type conversion successful.
```

In [ ]:
```
# checking for the correct data type of the variable

chocolate_data.dtypes
```

Out[188]:
```
Competitorname      category
Chocolate              int64
Fruity                 int64
Caramel                int64
Peanutyalmondy         int64
Nougat                 int64
Crispedricewafer       int64
Hard                   int64
Bar                    int64
Multiple pieces        int64
Sugarpercent         float64
Pricepercent         float64
dtype: object
```

5. Checking for outliers in the dataset

a. Boxplot

The below code creates **boxplots** for the various field values of the chocolate dataset in order to check for outliers present in the dataset. Here, the boxplots are implemented for the variables **Sugarpercent**, and **Pricepercent**, as shown in the figures below.

There are outliers present in the Sugarpercent column as observed in the boxplot below, which will not be removed as each of the data point is important for analysis and model building.

In [ ]:
```
# creating boxplot for 'Sugarpercent' and 'Pricepercent' variable

fig, axs = plt.subplots(1, 2, figsize=(10, 5))
axs[0].boxplot(chocolate_data['Sugarpercent'])
axs[1].boxplot(chocolate_data['Pricepercent'])
axs[0].set_title('Boxplot for Sugarpercent')
axs[1].set_title('Boxplot for Pricepercent')
axs[0].set_ylabel('Data Values')
axs[1].set_ylabel('Data Values')

plt.show()
```

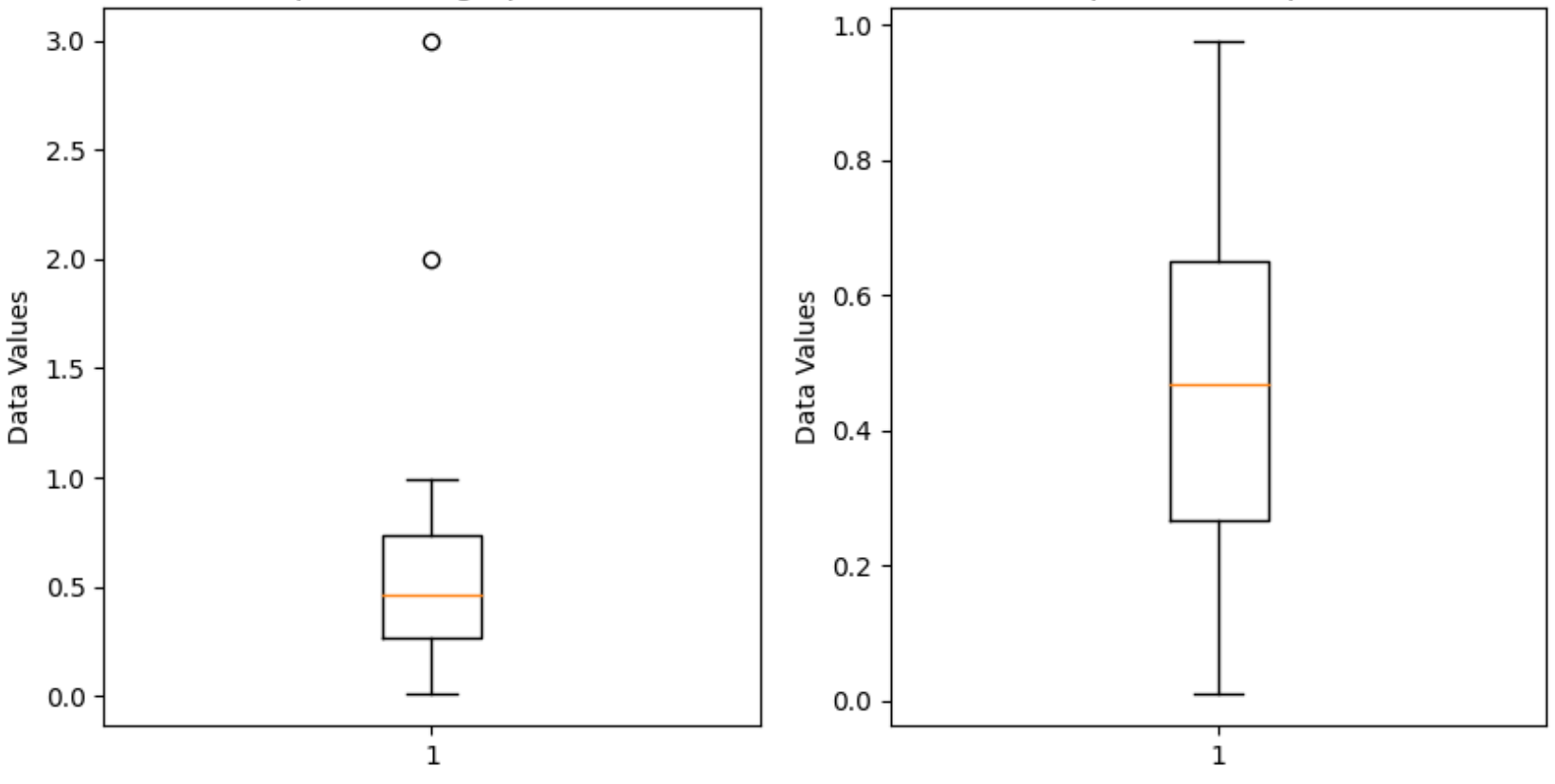

Figure 3. Boxplot for Sugarpercent & Pricepercent

b. Distribution Plot

The distribution plot for the various parameters of the dataset values gives an overview of the outliers that are present and the distribution of the data points across present in the dataset.

The plot below for *Chocolate*, *Sugarpercent*, *Fruity*, and *Nougat* shows that the data is **normally distributed** across the data points, meaning that the data points are evenly distributed around the mean value.

In [ ]:
```
# distribution plot for the Chocolate & Sugarpercent

plt.figure(figsize=(16,5))
plt.subplot(1,2,1)
sns.distplot(chocolate_data['Chocolate'])
plt.subplot(1,2,2)
sns.distplot(chocolate_data['Sugarpercent'])
plt.show()
```

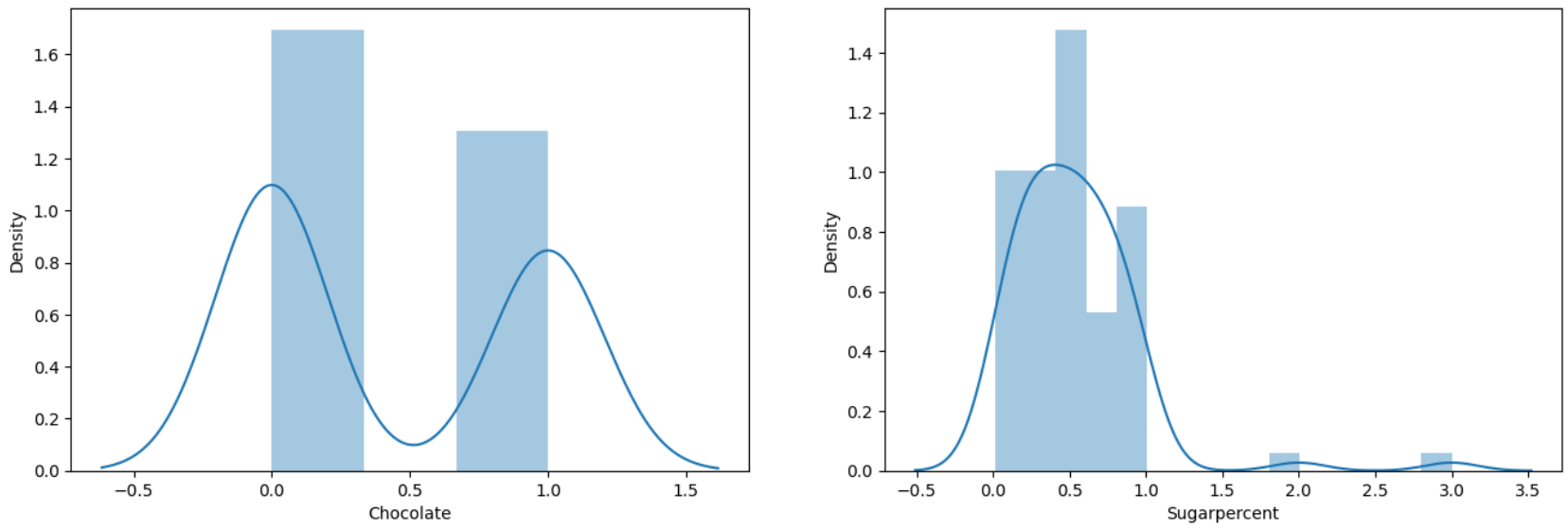

Figure 4. Distribution Plot for Chocolate & Sugarpercent

In [ ]:
```
# distribution plot for the Fruity & Nougat

plt.figure(figsize=(16,5))
plt.subplot(1,2,1)
sns.distplot(chocolate_data['Fruity'])
plt.subplot(1,2,2)
sns.distplot(chocolate_data['Nougat'])
plt.show()
```

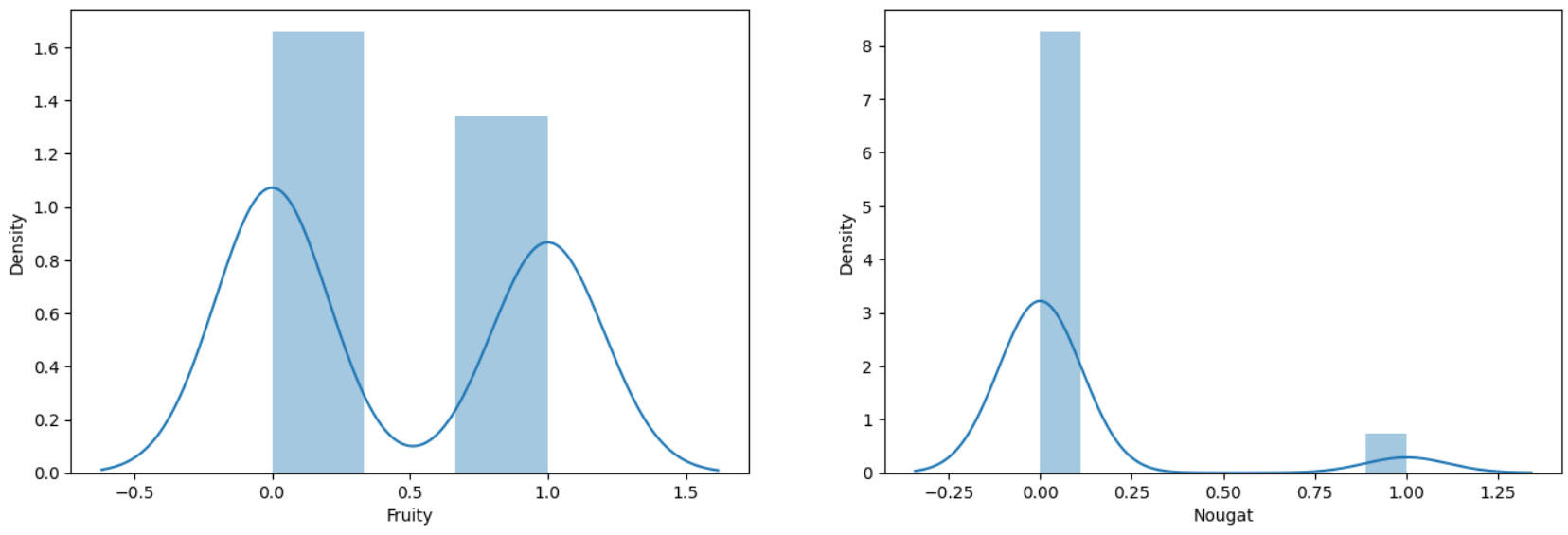

Figure 5. Distribution Plot for Fruity & Nougat

Step 3: Data Visualization

In [ ]:
```
# 1. Price Percent Analysis

plt.figure(figsize=(17,5))
sns.countplot(x='Pricepercent', data=chocolate_data, palette="pastel")
plt.title('\nPrice Percent Analysis')
plt.show()
```

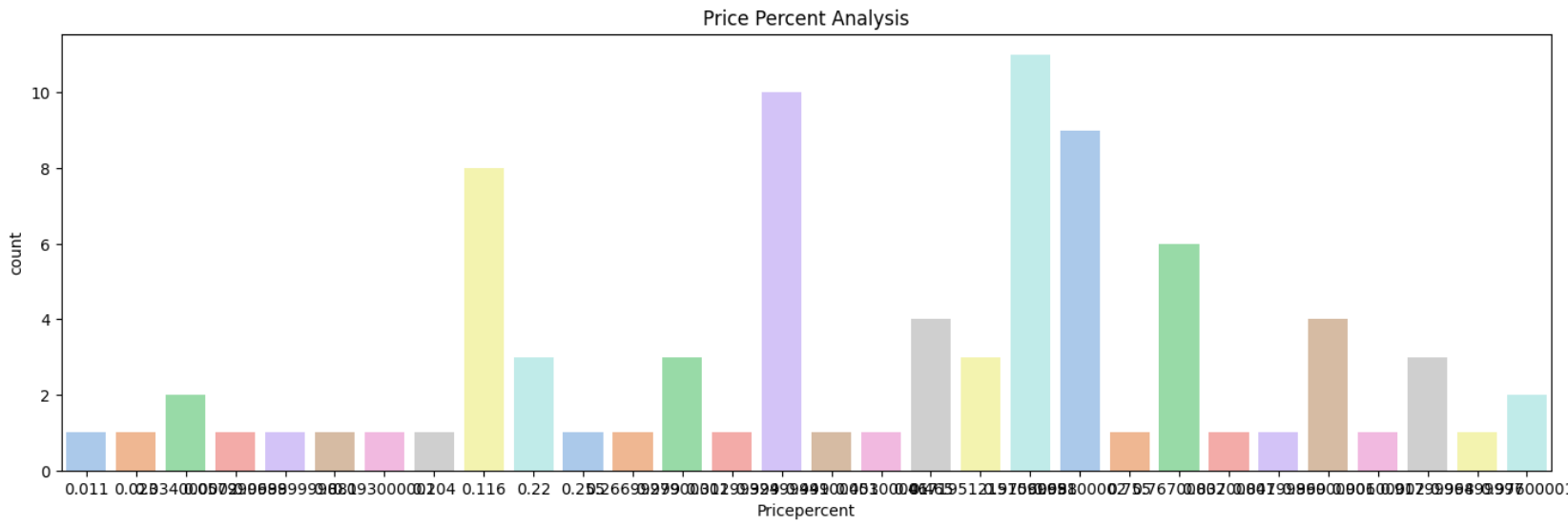

Figure 6. Pricepercent Count Analysis

The above visualization plots the **Price Percent graph** which helps in analyzing the price percent count of the data points, where the various count values of the percentage are displayed that can show which percent value has a higher count.

In [ ]:
```
# 2. Presence of Chocolate & Caramel (Yes or No)

fig, axs = plt.subplots(1, 2, figsize=(10, 5))
sns.countplot(x='Chocolate', data=chocolate_data, palette="pastel", ax=axs[0])
axs[0].set_title('\nPresence of Chocolate in Candy Bar')
sns.countplot(x='Caramel', data=chocolate_data, palette="pastel", ax=axs[1])
axs[1].set_title('\nPresence of Caramel in Candy Bar')
plt.show()
```

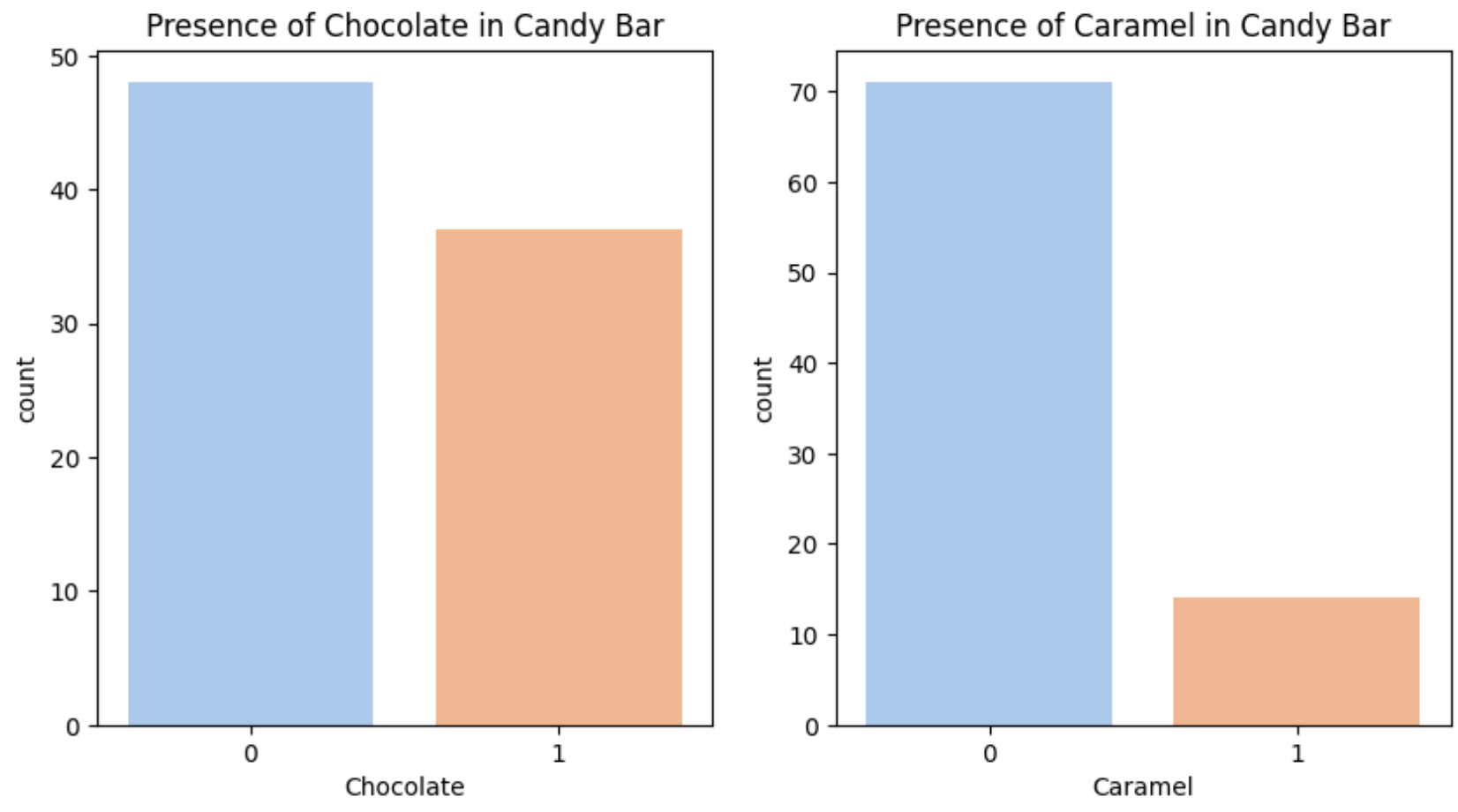

Figure 7. Presence of Chocolate & Caramel in candy bar

The plot for count of chocolate and caramel help in understanding that there are binary values present in the dataset and that both chocolate and caramel have highest count for value 0 as compared to value 1, which means that the data has maximum values for no chocolate and no caramel in candy bars.

In [ ]:
```
# 3. Multiple Pieces feature Analysis

plt.figure(figsize=(5,5))
sns.countplot(x='Multiple pieces', data=chocolate_data, palette="pastel")
plt.title('\nMultiple Pieces feature Analysis')
plt.show()
```

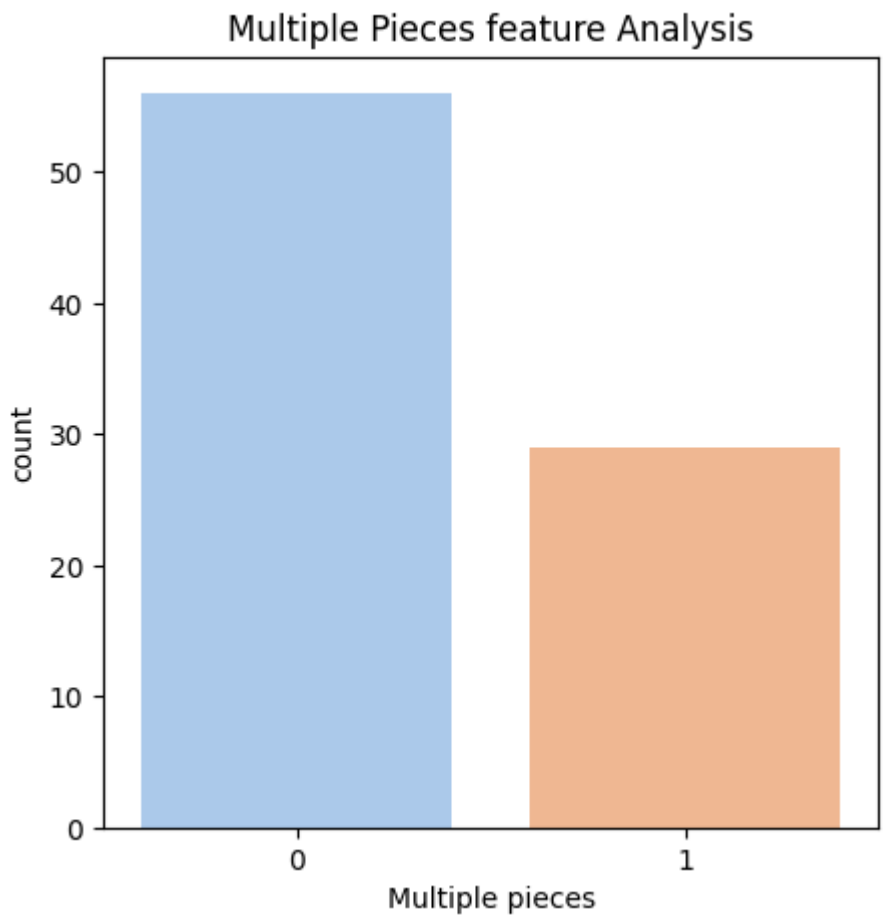

Figure 8. Multiple pieces feature analysis

The 'Multiple pieces' feature graph also helps in analyzing the data points which show that the candy bar have maximum count of no multiple pieces present in the dataset. Thus, the count of value 0 is more as compared to value 1 for the multiple pieces.

In [ ]:
```
# 4. Sugar Percent Analysis

plt.figure(figsize=(17,5))
sns.countplot(x='Sugarpercent', data=chocolate_data, palette="pastel")
plt.title('\nSugar Percent Analysis')
plt.show()
```

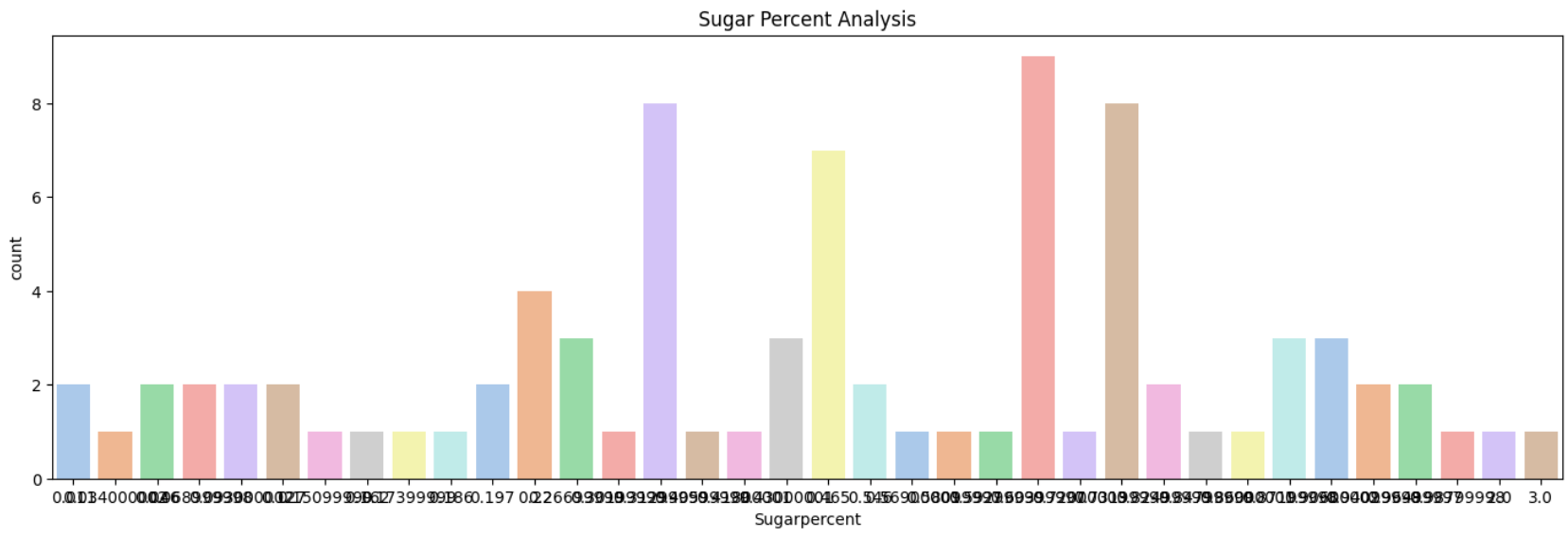

Figure 9. Sugarpercent Count Analysis

The above visualization plots the Sugar Percent graph which helps in analyzing the sugar percent count of the data points, where the various count values of the percentage are displayed that can show which percent value has a higher count indicating the sugar percent present in the candy bar.

Step 4: Pre-Modeling Steps

1. Feature Selection & Extraction
2. Label Encoding
3. Correlation Plot
4. Defining the features for model training
5. Spliting the dataset into train & test set
6. Standardization

1. Feature Selection and Extraction

In [ ]:
```
# Feature Extraction

target = 'Pricepercent'

features, train = featurewiz(chocolate_data, target, corr_limit=0.7, verbose=2, sep=",",
header=0,test_data="", feature_engg="", category_encoders="")
```

```
#######################################################################################
############      F A S T  F E A T U R E  E N G G  A N D  S E L E C T I O N ! ########
# Be judicious with featurewiz. Don't use it to create too many un-interpretable features! #
#######################################################################################
featurewiz has selected 0.7 as the correlation limit. Change this limit to fit your needs...
Skipping feature engineering since no feature_engg input...
Skipping category encoding since no category encoders specified in input...
#### Single_Label Regression problem ####
    Loaded train data. Shape = (85, 12)
    Some column names had special characters which were removed...
#### Single_Label Regression problem ####
No test data filename given...
#######################################################################################
######################## C L A S S I F Y I N G  V A R I A B L E S  ####################
#######################################################################################
        1 variable(s) to be removed since ID or low-information variables
        variables removed = ['Competitorname']
GPU active on this device
    Tuning XGBoost using GPU hyper-parameters. This will take time...
```

The above code generated a feature selection & extraction report using the '**featurewiz**' function that helped in understanding which features are to be taken into consideration for the prediction of the price of the candy bar where the target variable is the '**Pricepercent**' column.

The features selected by the featurewiz function are as shown below.

In [ ]:
```
print("The extracted features are:")
features
```

```
The extracted features are:
```

Out[207]:
```
['Bar', 'Chocolate', 'Sugarpercent']
```

2. Label Encoding

Since some of the selected features for prediction are categorical data and most of the ML models require the data to be numerical or binary, it is important that these features are converted into **binary** or **numerical** type data in order for the model to be able to classify the classes.

Label Encoding is a method which helps to **convert the categorical variables** into **numerical values**, thus helping to transform the data point into a format where the algorithm is able to process the data for classification. *LabelEncoder()* function is used to encode the categorical type data to numerical type, where new columns of data are created for the categorical field value in the dataset which will be used in the training of the model.

In [8]:
```
labelencoder = LabelEncoder()

chocolate_data['Competitorname_Label'] = labelencoder.fit_transform(chocolate_data["Competitorname"])
print("Label Encoding successful.")
```

```
Label Encoding successful.
```

In [ ]:
```
# checking for data values and field names after Label Encoding

chocolate_data
```

Out[209]:

	Competitorname	Chocolate	Fruity	Caramel	Peanutyalmondy	Nougat	Crispedricewafer	Hard	Bar	Multiple pieces	Sugarpercent	Pricep
0	100 Grand	1	0	1	0	0	1	0	1	0	0.732	
1	3 Musketeers	1	0	0	0	1	0	0	1	0	0.604	
2	One dime	0	0	0	0	0	0	0	0	0	0.011	
3	One quarter	0	0	0	0	0	0	0	0	0	0.011	
4	Air Heads	0	1	0	0	0	0	0	0	0	0.906	
...	...	...	...	...	...	...	...	...	...	...	...	
80	Twizzlers	0	1	0	0	0	0	0	0	0	0.220	
81	Warheads	0	1	0	0	0	0	1	0	0	0.093	
82	WelchÕs Fruit Snacks	0	1	0	0	0	0	0	0	1	0.313	
83	WertherÕs Original Caramel	0	0	1	0	0	0	1	0	0	0.186	
84	Whoppers	1	0	0	0	0	1	0	0	1	0.872	

85 rows × 13 columns

Table 7. Chocolate Data after Label Encoding

In [ ]:
```
chocolate_data.columns
```

Out[210]:
```
Index(['Competitorname', 'Chocolate', 'Fruity', 'Caramel', 'Peanutyalmondy',
       'Nougat', 'Crispedricewafer', 'Hard', 'Bar', 'Multiple pieces',
       'Sugarpercent', 'Pricepercent', 'Competitorname_Label'],
      dtype='object')
```

Q6. Build a correlation plot. In 2-3 sentences, discuss the results and if there are any strong relationships. Anything we need to be concerned about from a multicollinearity standpoint?

A6. The correlation matrix is as plotted below and the relationship between the variables of the dataset are further discussed based on the results from the correlation plot and the VIF Score.

3. Corelation Plot

A **correlation plot** or matrix is a *visual representation of the variables* present in the dataset which helps in understanding the *relationship* between the different variables and how highly the variables are corelated to each other.

The values of the correlation plot range from **-1 to 1**, where -1 indicates a **negative correlation** between the variables, 0 indicates **no correlation**, and 1 indicates a **positive correlation**.

The variables that have positive correlation are said to be highly correlated to each and hence either of the two variables must be removed for the model building as it may lead to **multicollinearity** where the efficiency of the model may reduce.

In [11]:
```
# plotting correlation matrix

plt.figure(figsize = (12,5))
ax = plt.subplot()
sns.heatmap(chocolate_data.corr(),annot=True, fmt='.1f', ax=ax, cmap="Blues")
ax.set_title('Correlation Plot');
```

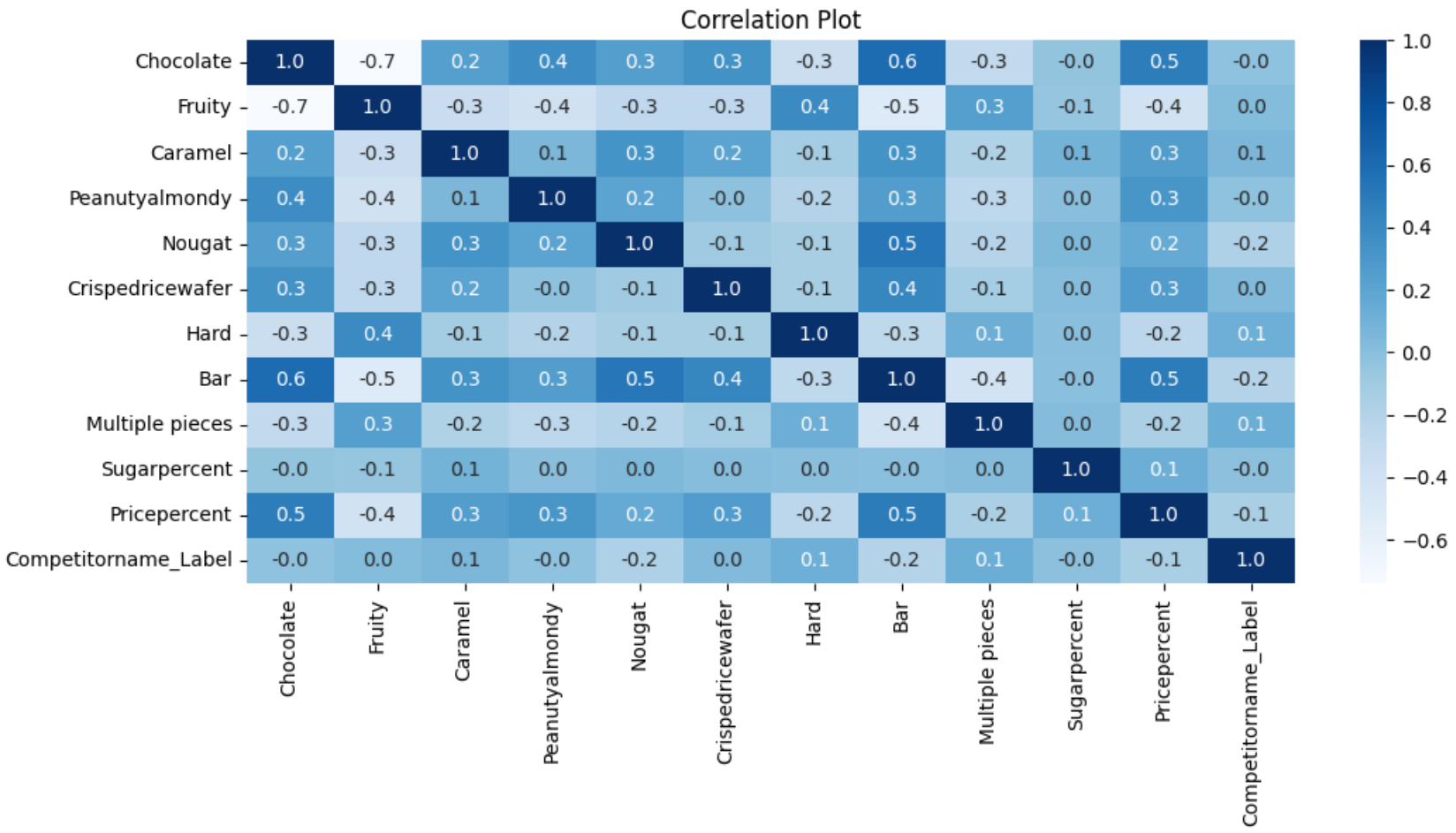

Figure 10. Correlation Matrix

A6.

Looking for strong corelations between independent & dependent variables

---

As observed in the correlation matrix above, we see that there are many variables or features that are *highly correlated* to each other and hence we need to analyze the features that are strongly correlated such that these features are excluded from the training of the model in order to avoid **multicollinearity** and *improve the efficiency of the model*. The following features are strongly correlated with the other features in the dataset and can be excluded from model building to avoid multicollinearity.

Correlation among the variables:

1. **Chocolate** has high correlation with **Bar and Peanutalmondy** with correlation value 0.6 and 0.4 respectively
2. **Fruity** has high correlation with **Hard** with correlation value of 0.4
3. **Nougat** has high correlation with **Bar** with correlation value of 0.5
4. **Crispedricewafer** has high correlation with **Bar** with correlation value of 0.4
5. **Chocolate**, and **Bar** have **high positive influence on Pricepercent** with correlation value of 0.5 whereas **Caramel, Peanutyalmondy, Crispedricewafer have moderate positive influence on Pricepercent** with value 0.3. **Hard & Fruity** have **negative influence on the Pricepercent** of the candy bar.

Thus, due to the **high correlation** values between the variables, multicollinearity may occur leading to the inefficiency of the model and hence the highly correlated variables need to be excluded from the training of the model in order to improve the performance of the model and predict the pricepercent accurately.

Checking VIF score for Multicollinearity

VIF score i.e., **Variance Inflation Factor** is a *measure of multicollinearity* between the independent variables in the regression analysis. This calculates the variance of the variables which helps in understanding the coefficient value and how much the variable is inflated due to collinearity in the model. The VIF score from **range 0 to 5** can be accepted to be considered for the training of the model, while values above 5 are considered to have high multicollinearity which would affect the accuracy and performance of the model, hence should be excluded.

In [ ]:
```
# checking VIF score for field values

numerical_chocolate_data = chocolate_data.select_dtypes(include=[np.number])
vif_score1 = pd.DataFrame()
vif_score1["Feature"] = numerical_chocolate_data.columns
vif_score1["VIF Score"] = [variance_inflation_factor(numerical_chocolate_data.values, i) for i in range(numer
print(vif_score1)
```

```
                Feature  VIF Score
0             Chocolate   4.205717
1                Fruity   2.910076
2               Caramel   1.611366
3         Peanutyalmondy   1.615344
4                Nougat   1.989995
5       Crispedricewafer   1.699454
6                  Hard   1.471485
7                   Bar   4.031943
8        Multiple pieces   1.847196
9           Sugarpercent   2.452188
10          Pricepercent   5.494272
11  Competitorname_Label   3.540817
```

Table 8. VIF Score Table

From the above code and results, it is observed that the VIF score for all the independent variables is below the range of 5, which indicates that these variables can be considered for the training of the model and thus there are less chances of multicollinearity to occur.

a. Understanding the top features selected by Correlation Matrix

In [ ]:
```
corr_result = chocolate_data.corr()
correlation_price = corr_result['Pricepercent'].sort_values(ascending=False)
topfeatures = correlation_price[1:6]
print("The top features selected by correlation matrix are:\n")
print(topfeatures)
```

```
The top features selected by correlation matrix are:

Bar                 0.490398
Chocolate           0.483955
Peanutyalmondy      0.314518
Crispedricewafer    0.268633
Caramel             0.263344
Name: Pricepercent, dtype: float64
```

b. Lasso Regression to select the most important features for model training

In [ ]:
```
# creating a new dataframe excluding categorical variable

new_chocolate_data = pd.DataFrame()
new_chocolate_data = chocolate_data.drop(['Competitorname'], axis=1)
```

In [ ]:
```
A = new_chocolate_data.drop(['Pricepercent'], axis=1)
B = new_chocolate_data['Pricepercent']
lasso_result = Lasso(alpha=0.1)
lasso_result.fit(A, B)
coef = pd.Series(lasso_result.coef_, index=A.columns)
features_lasso = coef.abs().sort_values(ascending=False).head(4).index
print("The top three features selected by Lasso regression:\n")
print(features_lasso)
```

```
The top three features selected by Lasso regression:

Index(['Competitorname_Label', 'Chocolate', 'Fruity', 'Caramel'], dtype='object')
```

c. Features selected for model building

The features that are selected for the model building based on the Feature Selection & Extraction, Correlation Plot values, Lasso Regression, and VIF Score are as follows:

Feature Selection & Extraction	Correlation Plot Values	Lasso Regression	Features with VIF Scores in range 0 to 5
Bar	Bar	Competitorname	Competitorname
Chocolate	Chocolate	Chocolate	Chocolate
Sugarpercent	Peanutyalmondy	Fruity	Fruity
	Crispedricewafer	Caramel	Caramel
	Caramel		Peanutyalmondy
			Nougat
			Crispedricewafer
			Hard
			Bar
			Multiple pieces
			Sugarpercent

Table 9. Features selected for model building

4. Defining the features for model training (Dimensionality Reduction)

The model is trained & built on the below mentioned features that are selected from the analysis of the Feature selection and extraction report, Correlation matrix, Lasso Regression, and VIF score for multicollinearity.

1. Competitorname & Multiple pieces are not considered as relevant features for model building as they do not have a strong influence in the prediction of the price percent, hence excluded
2. Fruity & Hard are highly correlated, hence excluded from the training of the model

In [9]:
```
X = chocolate_data[['Chocolate', 'Caramel', 'Peanutyalmondy', 'Nougat', 'Crispedricewafer', 'Bar','Sugarperce
y = chocolate_data['Pricepercent']
```

5. Splitting the dataset into train & test set

The dataset is split into training and testing data with a random split of **80%** train set and **20%** for test data.

In [10]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state = 42)
```

6. Standardization

Standardization is performed on the split dataset in order to make the features selected comparable to a standardized scale.

In [11]:
```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
print("Standardization successful")
```

```
Standardization successful
```

Step 5: Model Building

Q7. Model the dataset to predict pricepercent (the higher the value, the more expensive the candy bar is). Which three variables have the lowest p-value (closest to 0)? If there are more than three (tie), rank the top three by the highest coefficient (by absolute value). For this response, please provide the variables, p-values, and coefficients (in absolute and non-absolute form). Present these findings in a table and document your steps in 2-3 sentences.

A7. Building the **Linear Regression** model to predict the price of the candy bar based on the features selected for training and answering the above questions.

Fitting the Linear Regression model

In [12]:
```
regressor_model = LinearRegression()
regressor_model.fit(X_train_scaled, y_train)
```

Out[12]:
```
LinearRegression
LinearRegression()
```

Displaying the coefficients & intercepts after fitting the model

---

As observed from the below code, the coefficient values of the variables are either positive or negative, which indicates that the variables with **positive** value have a **positive relationship with the target variable** whereas values having a **negative sign** indicate that there is a **negative relationship between the independent variable and the target variable.** This helps in understanding which feature contributes in the prediction of the target variable, which in this case is this '**Pricepercent**'.

In [30]:
```
coefficients = pd.DataFrame(regressor_model.coef_, X.columns, columns=['Coefficient'])
print(coefficients)
print('Intercept: ', regressor_model.intercept_)
```

```
                  Coefficient
Chocolate            0.070642
Caramel              0.036424
Peanutyalmondy       0.035875
Nougat              -0.039819
Crispedricewafer    -0.000231
Bar                  0.093012
Sugarpercent         0.019817
Intercept:  0.4644056680283357
```

In [31]:
```
# significant features

significant_features= coefficients.nlargest(3, 'Coefficient')
print("The top 3 significant features with the coefficient values are:\n")
print(significant_features)
```

```
The top 3 significant features with the coefficient values are:

           Coefficient
Bar           0.093012
Chocolate     0.070642
Caramel       0.036424
```

Significant Variable Selection

The variable selection metrics is used to identify the most significant variable or relevant features that needs to be included in the training of the Linear Regression model. From the coefficients and p-values of the independent variables, and the standard error of the model, it is observed that **'Experience' and 'Mortgage'** are **not significant variables** and do not contribute in the classification of the loan approval model.

In [32]:
```
X2 = sm.add_constant(X)
linearreg_model1 = sm.OLS(y, X2)
model_result = linearreg_model1.fit(method='pinv')
p_values = model_result.pvalues[1:]
standard_error = model_result.params[1:] / model_result.bse[1:]
significant_var = pd.DataFrame({'p-value': p_values, 'Standard Error': standard_error})
significant_var['Significant Variable'] = np.where(significant_var['p-value'] < 0.05, 'Yes', 'No')
print('\nSignificant Variable Selection Table:\n')
print(significant_var)
```

```
Significant Variable Selection Table:

                   p-value  Standard Error Significant Variable
Chocolate         0.070139        1.836529                   No
Caramel           0.240017        1.184097                   No
Peanutyalmondy    0.108485        1.623898                   No
Nougat            0.170254       -1.384337                   No
Crispedricewafer  0.998270       -0.002176                   No
Bar               0.014619        2.498075                  Yes
Sugarpercent      0.164980        1.401845                   No
```

Table 10. Significant variable selection

Summary Report of the Linear Regression model [5]

The summary report of the Linear Regression model displays various results and scores of the machine learning model that helps in understanding if the trained model is efficient or not, which in this case is the Linear Regression model for predicting the price percent of a candy bar.

1. *R-squared:* It is the coefficient of determination that basically indicates the proportion of variation in the dependent variables. The **higher R-squared** value indicates a good fit model, and as it is observed from the summary report below, the R-squared value is **0.736**, which indicates that the R-sqaured score is not high or close to 1, and thus requires improvement in the training of the model, as it might not be a good fit model.
2. *Adj. R-squared:* This is same as the R-squared value with a difference that while performing **multiple linear regression model** we consider the Adj. R-squared value as the addition of unnecessary variables to the model need a penalty and thus the R-squared value is adjusted for multiple features, else for a single independent variable, R-squared and Adj. R-squared value are the same. For the prediction of price percent, the Adj. R-squared value is **0.706**.
3. *AIC & BIC values:* These values are used for the model robustness and the aim here is to minimize the AIC and BIC values in order to get an efficient model.
4. *Durbin-Watson:* This measure provides statistics for *autocorrelation in the residual*, which means that if the residual values are autocorrelated then the model will be biased which should not be the case, meaning that no value should be depending upon the other value. The ideal value for the Durbin-Watson test is from **0 to 4**, and here the Durbin-Watson test value comes to be equal to **2.027**, which is under the ideal score range. [5]
5. *coef & P>|t|:* The p-value in the summary report helps to understand which independent variables are significantly important for consideration and which are not. They are used to determine the significance of the predictor variables in the model.

   a. If **p-value is less** than the significance values of 0.05, it is considered to be **statistically significant**

   b. If the **p-value is greater** than the significance values of 0.05, it is **not considered statistically significant** which indicates that it is not contributing to the prediction of the target variables

---

- From the summary report below, it is observed that from all the attributes and features considered for the training of the model, three of them have p-value less than 0.05, indicating that the attributes are statistically significant and are contributing to the prediction of the price percent. The features that are statistically significant are **Chocolate, Bar, and Sugarpercent.**
- Considering the statistically significant variables, there is a *positive coefficient* obtained for all the selected features which implies that these features will contribute in the prediction of the price percent of the candy bar, as they have a positive influence on the prediction. This means that if the sugarpercent is high, the price percent is higher, whereas if the candy bar has chocolate or is a bar implies a high percentage price of the candy.

In [33]:
```
model = sm.OLS(y_train, X_train).fit()
print("\t\t\t\tSummary Report")
print("-------------------------------------------------------------------------------------\n")
print(model.summary())
```

```
                                Summary Report
-------------------------------------------------------------------------------

                                 OLS Regression Results
================================================================================
Dep. Variable:           Pricepercent   R-squared (uncentered):                   0.736
Model:                            OLS   Adj. R-squared (uncentered):              0.706
Method:                 Least Squares   F-statistic:                              24.35
Date:                Sat, 29 Apr 2023   Prob (F-statistic):                    1.84e-15
Time:                        21:14:50   Log-Likelihood:                         -9.4450
No. Observations:                  68   AIC:                                      32.89
Df Residuals:                      61   BIC:                                      48.43
Df Model:                           7
Covariance Type:            nonrobust
================================================================================
                     coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
Chocolate            0.2997      0.091      3.281      0.002       0.117       0.482
Caramel              0.1284      0.104      1.230      0.223      -0.080       0.337
Peanutyalmondy       0.1386      0.106      1.309      0.195      -0.073       0.350
Nougat              -0.1674      0.169     -0.988      0.327      -0.506       0.171
Crispedricewafer    -0.0081      0.151     -0.053      0.958      -0.311       0.295
Bar                  0.2167      0.129      1.686      0.097      -0.040       0.474
Sugarpercent         0.2994      0.058      5.121      0.000       0.183       0.416
==============================================================================
Omnibus:                        7.622   Durbin-Watson:                   2.027
Prob(Omnibus):                  0.022   Jarque-Bera (JB):               14.950
Skew:                          -0.056   Prob(JB):                     0.000567
Kurtosis:                       5.294   Cond. No.                         5.71
==============================================================================

Notes:
[1] R² is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Q7. Which three variables have the lowest p-value (closest to 0)? If there are more than three (tie), rank the top three by the highest coefficient (by absolute value). For this response, please provide the variables, p-values, and coefficients (in absolute and non-absolute form). Present these findings in a table and document your steps in 2-3 sentences.

---

a. Variables with p-value

In [34]:
```
pvalue_sort = model.pvalues
table1 = pd.DataFrame({'p-value': pvalue_sort})
table1 = table1.sort_values('p-value')
print(table1)
```

```
                  p-value
Sugarpercent     0.000003
Chocolate        0.001711
Bar              0.096915
Peanutyalmondy   0.195324
Caramel          0.223239
Nougat           0.327109
Crispedricewafer 0.957693
```

b. The three variables that have the lowest p-value (closest to 0) are as follows.

In [35]:
```
table1 = table1.sort_values('p-value').head(3)
print(table1)
```

```
               p-value
Sugarpercent  0.000003
Chocolate     0.001711
Bar           0.096915
```

c. Ranking the top three variables by the highest coefficient (by absolute value).

In [36]:
```
coef_sort = abs(model.params).sort_values(ascending=False).head(3)
table2 = pd.DataFrame({'Coefficient (abs)': coef_sort})
table2 = table2.loc[coef_sort.index]
print(table2)
```

```
              Coefficient (abs)
Chocolate              0.299671
Sugarpercent           0.299415
Bar                    0.216710
```

d. Variables with p-values and coefficients values (absolute & non-absolute form)

In [42]:
```
variable_dataframe = pd.DataFrame({'p-value': pvalue_sort, 'Coefficient': coef_sort})
variable_dataframe['Coefficient (abs)'] = abs(variable_dataframe['Coefficient'])
variable_dataframe = variable_dataframe.nsmallest(3, ['p-value', 'Coefficient (abs)'], keep='all')
variable_dataframe = variable_dataframe.sort_values('Coefficient (abs)', ascending=[False])
variable_dataframe = variable_dataframe[['p-value', 'Coefficient', 'Coefficient (abs)']]
print('Significant Variables Table (sorted based on highest coefficient value)\n')
variable_dataframe
```

```
Significant Variables Table (sorted based on highest coefficient value)
```

Out[42]:

	p-value	Coefficient	Coefficient (abs)
Chocolate	0.001711	0.299671	0.299671
Sugarpercent	0.000003	0.299415	0.299415
Bar	0.096915	0.216710	0.216710

Table 11. Significant variables (p-value & coefficient value)

The table above helps in analyzing the variables and features that are significant for the prediction of the price percent of the candy bar. The highest coefficient value indicates that the feature has a strong impact on predicting the price of a candy.

Here, as observed the coefficient value for **Chocolate** is the highest, and thus if the candy has chocolate, the price percent of the candy will be higher. The second highest coefficient value is for **Sugarpercent**, followed by **Bar**.

Q8.a. Choose a metric to understand how the model did. Specify which metric you choose and what that tells you about the model (3-4 sentences)

A8.a.

There are various evaluation metrics that help to understand the goodness of fit of a model as well as how the model performed with respect to its test dataset.

R-squared value is one of the metric to evaluate the performance of a linear regression model where a higher R-squared value indicates that the model is performing better in predicting the dependent variable. In this scenario, the R-squared value helps to evaluate how well the model is predicting the price of candy bar, where it indicates how the features are correlated with pricepercent and that the model is able to predict the price of the candy bar.

Here, in this scenario, the R-squared value obtained from the summary report is **0.736**, which indicates that **73.6% of the price percent prediction** can be explained by the independent variables selected for training. This means that the model is able to predict the price of a candy bar and is relatively a good fit model, but still requires additional improvement as the score is not close to 1.

Model Testing

In [ ]:
```
y_pred = regressor_model.predict(X_test_scaled)
```

Evaluating the performance of the model

In [ ]:
```
print("Model Evaluation of Linear Regression.")
print('Mean Absolute Error:', round(metrics.mean_absolute_error(y_test, y_pred),1))
print('Mean Squared Error:', round(metrics.mean_squared_error(y_test, y_pred),1))
print('Root Mean Squared Error:', round(np.sqrt(metrics.mean_squared_error(y_test, y_pred)),1))
print("R-Squared value:", round(metrics.r2_score(y_test, y_pred),2))
```

```
Model Evaluation of Linear Regression.
Mean Absolute Error: 0.1
Mean Squared Error: 0.0
Root Mean Squared Error: 0.2
R-Squared value: 0.42
```

Residual Plot

In [ ]:
```
plt.scatter(y_pred, y_test - y_pred)
plt.xlabel('Predicted')
plt.ylabel('Residual')
plt.axhline(y=0, color='k', linestyle='--')
```

Out[395]:
```
<matplotlib.lines.Line2D at 0x7f0dd5cd5660>
```

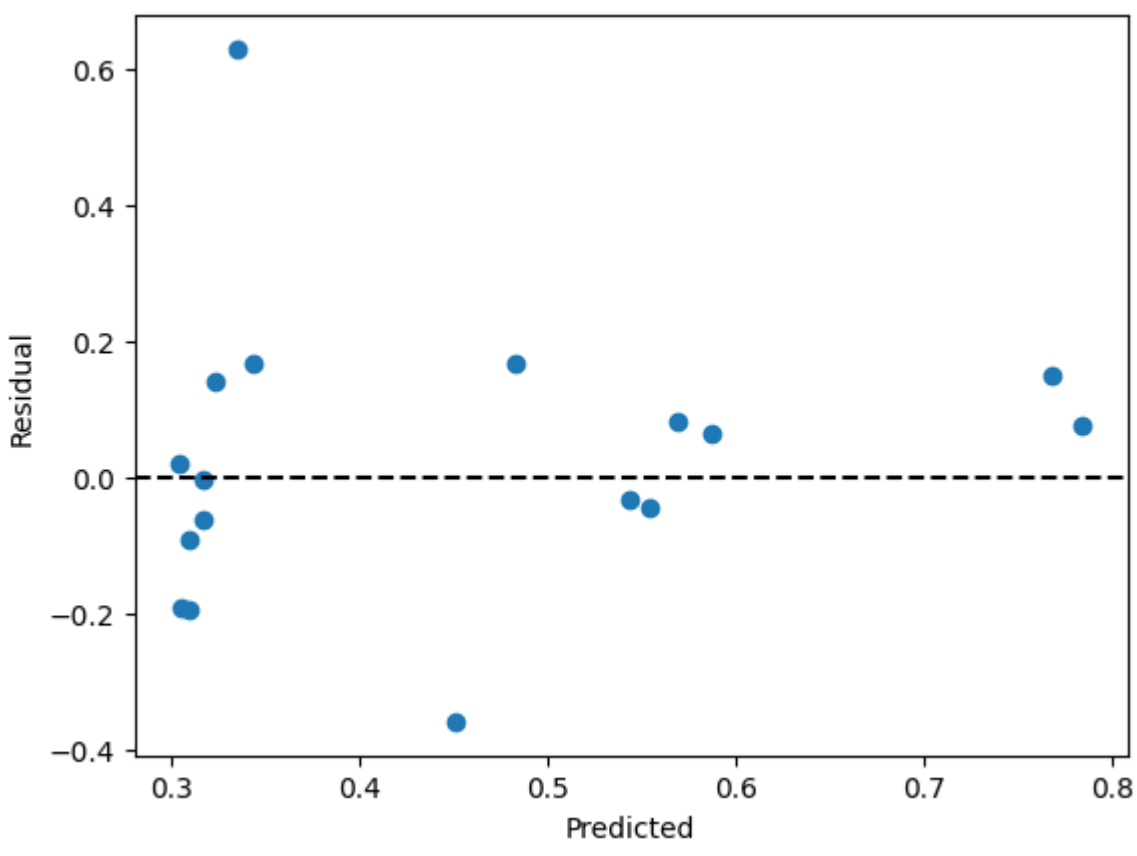

Figure 11. Residual Plot

In [ ]:
```
# accuracy of the model on training and testing set

print('Accuracy of Linear Regressor model on training set: {:.2f}'.format(regressor_model.score(X_train_scale
print('Accuracy of Linear Regressor model on test set:     {:.2f}'.format(regressor_model.score(X_test_scaled

result_LR = regressor_model.score(X_test_scaled, y_test)
result_LR = round(result_LR,4)
print("Overall Accuracy of the model is ",result_LR)
```

```
Accuracy of Linear Regressor model on training set: 0.34
Accuracy of Linear Regressor model on test set:     0.42
Overall Accuracy of the model is  0.4156
```

Overall Accuracy of the model is 41.56%

Accuracy on Training Data	Accuracy on Testing Data
34%	42%

The accuracy of both the train and test dataset is **very low** indicating that the model is **not efficient in predicting the price of the candy bar.**

The difference between the accuracy value for the train and test set is high indicating that the model is **underfitted** which can be due to the less amount of data values present in the dataset.

The performance of the model on the training and testing set indicates that the **model is not efficient and needs improvement** which may occur due to the overfitting or underfitting of the model. Thus, the quantity of data needs to be increased such that the model can accurately fit the train data.

Lastly, the **accuracy on the test data which is 42%** indicates that the model can accurately predict the price of the candy bar 42% of times, which means that there is still scope for improvement that can be done by adding **new features to the model and avoiding the issue of multicollinearity.**

Results

Q8.b. If you were Hershey (the candy company) making a candy bar and wanted to price it in the 60-70%-tile, which would be common characteristics of other candy bars in that price range. Does this match with what the model says drives the price of a candy bar up? 4-5 sentences

---

A8.b.

The summary report of the candy bar dataset above helps in identifying the features of candy bars that are priced in the range of 60-70 percentile. The **coefficient values** help to determine the **highest positive influence** on the price percent in the linear regression model.

As observed in the summary report, based on the features that have the highest coefficient values are **Chocolate, Sugarpercent, and Bar**. This indicates that the price percent of the candy bars is higher when there is high level of sugarpercent, and when the candy bar is chocolate and is a bar. Thus, the candy bars that are chocolate, bar, and have high sugar levels are priced higher and are likely to be priced in the range of 60-70 percentile range.

Therefore, the common charcteristics of other candy bars in that price range would be **Chocolate, Bar, and Sugarpercent** as they have the highest coefficient value, indicating a positive influence on the price percent of the candy bar.

In order to validate this result, the **features and characteristics can be compared which lie in the range of 60-70% range to the features that have high coefficient values obtained from the model**. Thus, we can conclude that if the features in the range of 60-70% range are similar to features having high coefficient values in the model then the linear regression model is able to accurately predict the price of the candy bar based on its features.

Conclusion

- **Linear Regression models**

  Linear regression models are the simplest method for modeling the relationship between the dependent variable and one or more independent variables which can be used to make predictions that will identify the most significant variable that contributes to the prediction of the target variable. These models assume a linear relationship between the independent and dependent variables and can be used to solve regression based problems.

- **Pricepercent Prediction using Linear Regression model**

  The price percent prediction for the candy bar is implemented using the linear regression model that helps to understand the features and parameters that are contributing in the price percent of the candy bar. The model built helps in analyzing the highest coefficient values that have a positive impact and influence on the price and thus can be considered while pricing the candy bar in a certain percentile range.

- **Model Performance & Accuracy**

  The model performance and accuracy is determined with the help of various evaluation metrics such as R-squared value, MSE, MAE, etc. which help in understanding if the model is accurate enough and is able to predict the price of the candy bar based on the features selected for training. As observed above, we see that the R-squared value obtained is 0.42 which is not a close value to 1 indicating that the model is not accurate enough in predicting the price of the candy. **Accuracy of 34% on train data and 42% on test data** indicates that the model is not performing well in predicting the price percent of the candy bar, which may be due to the **overfit model of the training data** which is not able to *predict on the new and unseen data points*. This results in poor performance of the model for predictions on new data points and thus it is necessary that the model's performance is improved to order to reevaluate the efficiency of the model.

- **Recommendations**
  - Based on the analysis and results, we conclude that the model is **underfitted as the accuracy values for training and testing data differ by a certain percent value**, and hence the model requires further improvement such that it can be used in future for further prediction of the candy bar price, which can be done by updating new features and data points that will increase the efficiency and performance of the model, implying a best-fit model for training.
  - The results obtained from the model and the summary recommends that the **price percent of the candy bar increases due to high level of sugarpercent and if the candy has chocolate and is a bar.** This is analyzed by comparing the coefficient values, and as these variables have the highest positive coefficient value, it indicates that they have a positive influence on the price of the candy bar. **Hence, the features that should be focused upon are 'Chocolate', 'Bar', and 'Sugarpercent'.**
  - Overall **model accuracy is inefficient** which indicates that the Linear Regression model is not able to predict the price of the candy bar accurately due to the **quantity and quality of training data**, hence the model requires improvement in order to increase the performance for prediction.

- **Future Scope**

  The model is able to predict the price of the candy bar and also extract the features that contribute to the prediction, but the performance and efficiency of the model is poor which means that the model is not able to accurately predict the pricepercent of the candy bar and thus requires reevaluating the performance of the model by adding new features to the model and increasing the training data. Based on the results, it is observed that the data is underfitted and also has multicollinearity issue, hence it is important that both the quantity and quality of data is improved to increase the efficiency of the model.

  Thus, the model can be improved and updated based on new features being added to the dataset that can help to better analyze the target variable.

References

[1] Keita, Z. (2023). Top Techniques to Handle Missing Values Every Data Scientist Should Know. https://www.datacamp.com/tutorial/techniques-to-handle-missing-data-values (https://www.datacamp.com/tutorial/techniques-to-handle-missing-data-values)

[2] Interpret the key results for Fit Regression Model - Minitab. (n.d.). (C) Minitab, LLC. All Rights Reserved. 2023. https://support.minitab.com/en-us/minitab/21/help-and-how-to/statistical-modeling/regression/how-to/fit-regression-model/interpret-the-results/key-results/ (https://support.minitab.com/en-us/minitab/21/help-and-how-to/statistical-modeling/regression/how-to/fit-regression-model/interpret-the-results/key-results/)

[3] Kumar, A. (2023, April 4). Mean Squared Error or R-Squared - Which one to use? - Data Analytics. Data Analytics. https://vitalflux.com/mean-square-error-r-squared-which-one-to-use/ (https://vitalflux.com/mean-square-error-r-squared-which-one-to-use/)

[4] Mali, K. (2022). Everything you need to Know about Linear Regression! Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/10/everything-you-need-to-know-about-linear-regression/ (https://www.analyticsvidhya.com/blog/2021/10/everything-you-need-to-know-about-linear-regression/)

[5] Mahmood, M. S. (2022, April 24). Simple Explanation of Statsmodel Linear Regression Model Summary. Medium. https://towardsdatascience.com/simple-explanation-of-statsmodel-linear-regression-model-summary-35961919868b (https://towardsdatascience.com/simple-explanation-of-statsmodel-linear-regression-model-summary-35961919868b)