Lab 3

Stepan Ivanov

Data Mining I

28th October 2018

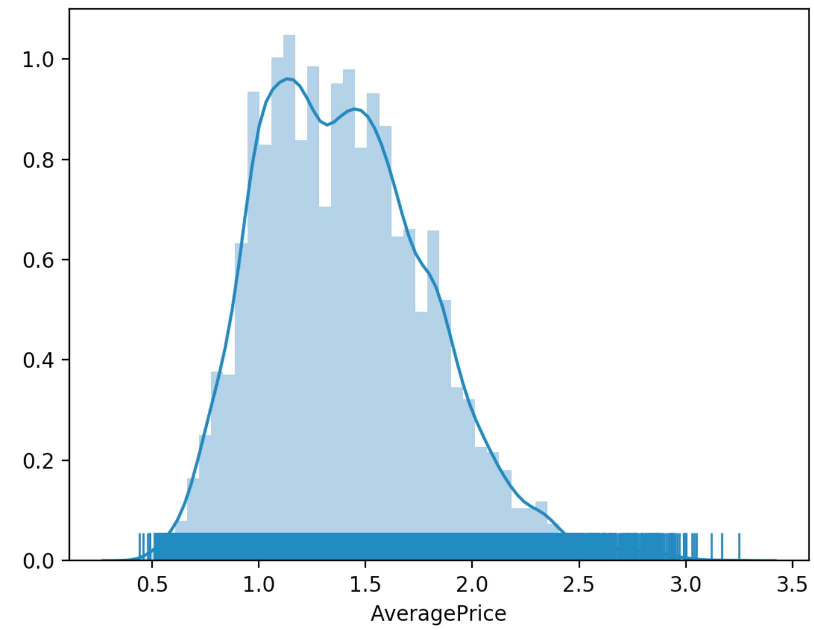
Fritz Gerald Santos
20071068
BSc (Hons) in Computer Forensics and Security

Table of Contents

See Appendices for the Python codes

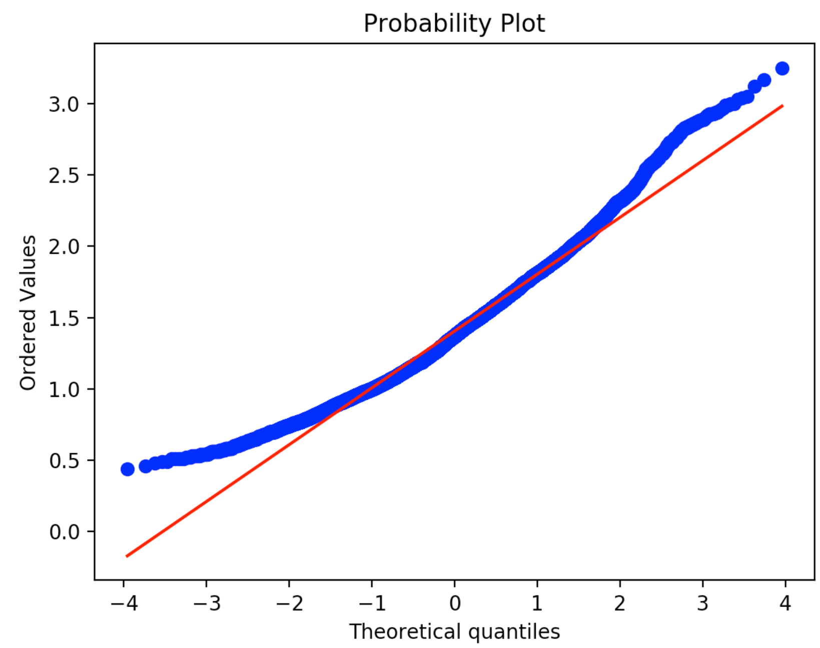
Normality

To test for normality, the first indicator is the visual representation of the data.



Appendix 1

A “bell-shaped” curve indicates that the data is normal. As it shows in the diagram above, the shape is almost perfect. To confirm the normality of the data, a quantile-quantile (Q-Q) plot can show if the quantiles of data matches against normal. If it does, it should be linear.



Appendix 2

Independence

Followed blog from Coding Disciple (2018) [1].

I will be considering two variables: the region where the avocados were sold and the average price of avocados.

Alternative hypothesis: the average price of avocados depends on the region where it was sold.
Null hypothesis: the average price is independent of the region where it was sold.

I first put the two variables in to a frequency table (appendix 3) then put the observed values into an array using the NumPy module (appendix 4). With this I am able to do the chi-squared test for independence (appendix 5).

Result for appendix 5: (25587.554859476208, 0.0, 13986)

Parameter 1 – the result of the chi-squared test	= 25587.554859476208
Parameter 2 – the P value	= 0.0
Parameter 3 – the Degree of Freedom	= 13986

A P value that is less than 0.05 tells us that we can reject the null hypothesis, meaning that there is a relationship between the prices of the avocados and the region where it was sold but we don't know what the relationship is.

Regression

This section was taken from the guidance of Alice ChiaHui Lui (2018)[2] on Kaggle.com.
The script for this section can be found at Appendix 6.
To create a regression model, the following steps must be taken:

1) Import required libraries
2) Clean the data
3) Create the two types into dummy variables and include them into the data frame.
4) Convert date time into quarters
5) Split the data frame into x and y
6) Create x and y trains using the scikit learn module
7) Use the x and y trains to create the model.

[1] Coding Disciple (2018). *'Chi-Squared Test for Independence in Python'* [Online]. Available at: https://codingdisciple.com/chi-squared-python.html (Accessed on: 25 October 2018).
[2] Lui, A.C. (2018) *'Avocado_Exploratory and Regression'* [Online]. Available at: https://www.kaggle.com/chiahuiliu/avocado-exploratory-and-regression/versions (Accessed: 27 October 2018).

Imports

```
import pandas
from sklearn.model_selection import train_test_split
import statsmodels.api as sm
from sklearn.metrics import explained_variance_score
```

Clean the Data

Convert the Date values into datetime using Pandas.

```
dataframe["Date"] = pandas.to_datetime(dataframe["Date"])
```

Change the data type of Region to categorical

```
dataframe['Region'] = dataframe['Region'].astype('category')
dataframe['Region'] = dataframe['Region'].cat.codes
```

cat.codes will convert any missing data as -1 (from Pandas documentation - Categorical Data).

Create Dummy Variables for Type

Dummy variables are artificial variables created to show variables with two or more distinct categories. It's used to trick the regression algorithm into appropriately evaluating attribute variables [3].

In this case the two values of the data series Type – conventional and organic are make into their own series where if the entry is conventional it will have a value of 1 and organic the value of 0, and vice versa.

```
dummy_type = pandas.get_dummies(dataframe['Type'])
```

The two series will be concatenated into the data frame.

```
dataframe = pandas.concat([dataframe, dummy_type], axis=1)
```

Convert Datetime into Quarters

This will divide the dates into quarters of the year.

```
dataframe['Date_Q'] = dataframe['Date'].apply(lambda x: x.quarter)
```

Split the Data Frame into X and Y

The Y column will take the Average Price series and the X column the rest.

```
X_columns = ['Total Volume', '4046', '4225', '4770', 'Total Bags', 'Small Bags', 'Large Bags',
'XLarge Bags', 'Year','Region', 'conventional', 'organic', 'Date_Q']
X = dataframe[X_columns]
Y = dataframe['AveragePrice']
```

---

[3] Skrivanek, S. (2009). *'The Use of Dummy Variables in Regression Analysis'* [Online]. Available at: https://www.moresteam.com/whitepapers/download/dummy-variables.pdf (Accessed on: 28 October 2018).

Create the X and Y Trains using the SciKit Learn Library.

The X and Y train will be used by the model.

X_train, y_train = train_test_split(X, Y, test_size=0.33, random_state=2018)

Use the X and Y Trains to Create the Model.

model = sm.OLS(y_train, X_train)
res = model.fit()
print(res.summary())

Which will result in:

```
                            OLS Regression Results
==============================================================================
Dep. Variable:           AveragePrice   R-squared:                       0.443
Model:                            OLS   Adj. R-squared:                  0.442
Method:                 Least Squares   F-statistic:                     809.3
Date:                Sun, 28 Oct 2018   Prob (F-statistic):               0.00
Time:                        19:49:24   Log-Likelihood:                -2644.7
No. Observations:               12226   AIC:                             5315.
Df Residuals:                   12213   BIC:                             5412.
Df Model:                          12
Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
[Total Volume -7.712e-05  4.78e-05     -1.614      0.106      -0.000    1.65e-05
4046           7.704e-05  4.78e-05      1.612      0.107   -1.66e-05       0.000
4225           7.723e-05  4.78e-05      1.616      0.106   -1.64e-05       0.000
4770           7.677e-05  4.78e-05      1.607      0.108   -1.69e-05       0.000
Total Bags       -0.0208     0.043     -0.480      0.631      -0.106       0.064
Small Bags        0.0209     0.043      0.481      0.630      -0.064       0.106
Large Bags        0.0209     0.043      0.481      0.630      -0.064       0.106
XLarge Bags       0.0209     0.043      0.481      0.630      -0.064       0.106
Year              0.0540     0.003     17.892      0.000       0.048       0.060
Region            0.0003     0.000      1.525      0.127   -7.85e-05       0.001
conventional   -107.8018     6.082    -17.724      0.000    -119.724     -95.880
organic        -107.3117     6.082    -17.644      0.000    -119.234     -95.390
Date_Q            0.0658     0.002     27.429      0.000       0.061       0.071
==============================================================================
Omnibus:                      488.861   Durbin-Watson:                   1.972
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              811.030
Skew:                           0.348   Prob(JB):                    7.71e-177
Kurtosis:                       4.052   Cond. No.                     1.30e+10
[==============================================================================
```

Appendices

Appendix 1 – distplot

```
seaborn.distplot(dataframe.AveragePrice, kde=True, rug=True)
```

Appendix 2 – probplot

```
stats.probplot(dataframe.AveragePrice, dist="norm", plot=pyplot)
```

Appendix 3 – Contingency table

```
contingency_table = pandas.crosstab(dataframe.AveragePrice, dataframe.Region,
margins=True)
```

Appendix 4 – Observed data into an array

```
f_obs = np.array([contingency_table.values])
```

Appendix 5 – Chi-Squared Test for Independence using stats module

```
print(stats.chi2_contingency(f_obs)[0:3])
```

Appendix 6 – Regression Model

```
import pandas
from sklearn.model_selection import train_test_split
import statsmodels.api as sm
from sklearn.metrics import explained_variance_score

dataframe = pandas.read_csv("avocado.csv")
dataframe.rename(columns={'year': 'Year', 'region' : 'Region', 'type' : 'Type'}, inplace=True)

dataframe["Date"] = pandas.to_datetime(dataframe["Date"])

dataframe['Region'] = dataframe['Region'].astype('category')
dataframe['Region'] = dataframe['Region'].cat.codes

dummy_type = pandas.get_dummies(dataframe['Type'])
dataframe = pandas.concat([dataframe, dummy_type], axis=1)


dataframe['Date_Q'] = dataframe['Date'].apply(lambda x: x.quarter)

X_columns = ['Total Volume', '4046', '4225', '4770', 'Total Bags', 'Small Bags', 'Large Bags',
'XLarge Bags', 'Year','Region', 'conventional', 'organic', 'Date_Q']
X = dataframe[X_columns]
Y = dataframe['AveragePrice']

X_train, X_test, y_train, y_test  = train_test_split(X, Y, test_size=0.33, random_state=2018)

model = sm.OLS(y_train, X_train)
res = model.fit()
print(res.summary())
```