Copy of FinalProject1

December 12, 2022

1 CPSC 392 Final Project

Kylie Mullenex, Sreya Vadlamudi, Sarah Fieck

CPSC 392

https://www.kaggle.com/datasets/raghadalharbi/all-products-available-on-sephora-website https://www.canva.com/design/DAFUCGiad2c/AJZrB2ckW8NE95W8-mKjQg/edit?utm_content=DAFUCGiad2c&utm_campaign=designshare&utm_medium=link2&utm_source=sl

1.0.1 Youtube Links

Sarah - https://youtu.be/34q95w8zisg

Sreya - https://youtu.be/ApLTW0LpcU0

Kylie - https://youtu.be/Zak4IELeNFU

[43]:
```
# Imports

import warnings
warnings.filterwarnings('ignore')

import pandas as pd
import numpy as np
from plotnine import *

from sklearn.linear_model import LinearRegression,LogisticRegression, Lasso#␣
 ↪Logistic Regression Model
from sklearn.metrics import accuracy_score, confusion_matrix,␣
 ↪plot_confusion_matrix,f1_score, recall_score, plot_roc_curve,␣
 ↪precision_score, roc_auc_score
from sklearn.preprocessing import StandardScaler #Z-score variables
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score,␣
 ↪accuracy_score #model evaluation


from sklearn.model_selection import train_test_split # simple TT split cv
from sklearn.model_selection import KFold # k-fold cv
```

```python
from sklearn.model_selection import LeaveOneOut #LOO cv
from sklearn.model_selection import cross_val_score # cross validation metrics
from sklearn.model_selection import cross_val_predict # cross validation metrics

from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB, BernoulliNB, CategoricalNB
from sklearn.model_selection import train_test_split

from sklearn import metrics
from sklearn.preprocessing import StandardScaler #Z-score variables

from sklearn.model_selection import train_test_split # simple TT split cv
from sklearn.model_selection import KFold # k-fold cv
from sklearn.model_selection import LeaveOneOut #LOO cv
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.metrics import plot_confusion_matrix, plot_roc_curve

from sklearn.model_selection import GridSearchCV

from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor #␣
 ↪Decision Tree
from sklearn.model_selection import train_test_split, KFold

from sklearn.preprocessing import StandardScaler #Z-score variables

from sklearn.metrics import accuracy_score, confusion_matrix,␣
 ↪mean_squared_error, plot_confusion_matrix, roc_auc_score, recall_score,␣
 ↪precision_score
import matplotlib.pyplot as plt

from sklearn.cluster import KMeans
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split

from sklearn import metrics
from sklearn.preprocessing import StandardScaler #Z-score variables
from sklearn.neighbors import NearestNeighbors

from sklearn.pipeline import make_pipeline
from sklearn.compose import make_column_transformer

from sklearn.cluster import KMeans
from sklearn.mixture import GaussianMixture
from sklearn.cluster import AgglomerativeClustering
from sklearn.cluster import DBSCAN
from sklearn.metrics import silhouette_score, silhouette_samples
from sklearn.decomposition import PCA

import scipy.cluster.hierarchy as sch
from matplotlib import pyplot as plt

%precision %.7g
%matplotlib inline
```

1.1 Loading Data

```
[44]: sephora = pd.read_csv("https://raw.githubusercontent.com/KylieMullenex/cpsc392/
       ↪main/sephora_website_dataset.csv")
      sephora = sephora.dropna()
      sephora = sephora.reset_index()
```

1.2 Question 1 - Sreya

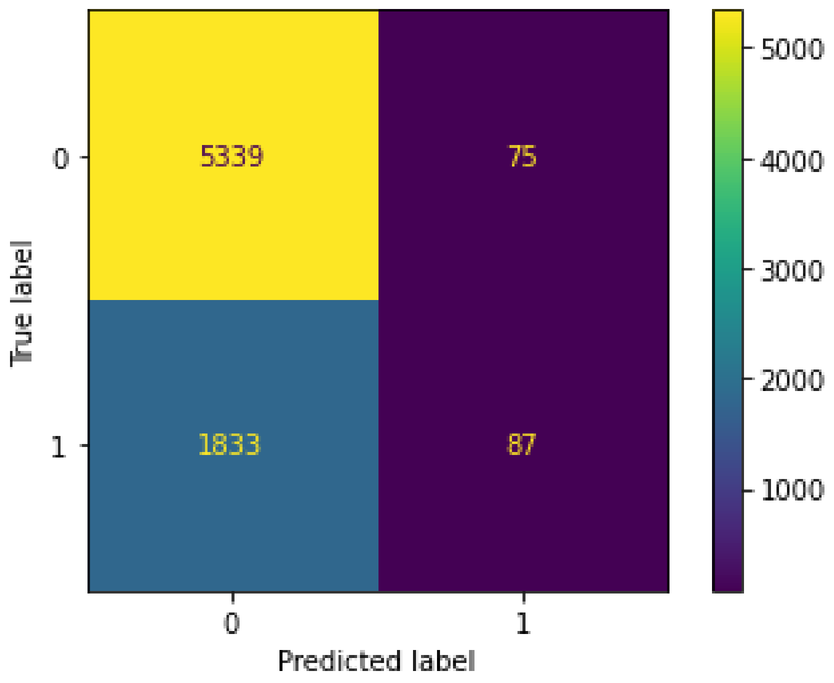
Using logistic regression, create a model in order to predict whether a Sephora product is exclusive or not based on the predictors rating, number of reviews, love, category, and price. Create a confusion matrix based on the results, and how does the train look in comparison to the test?

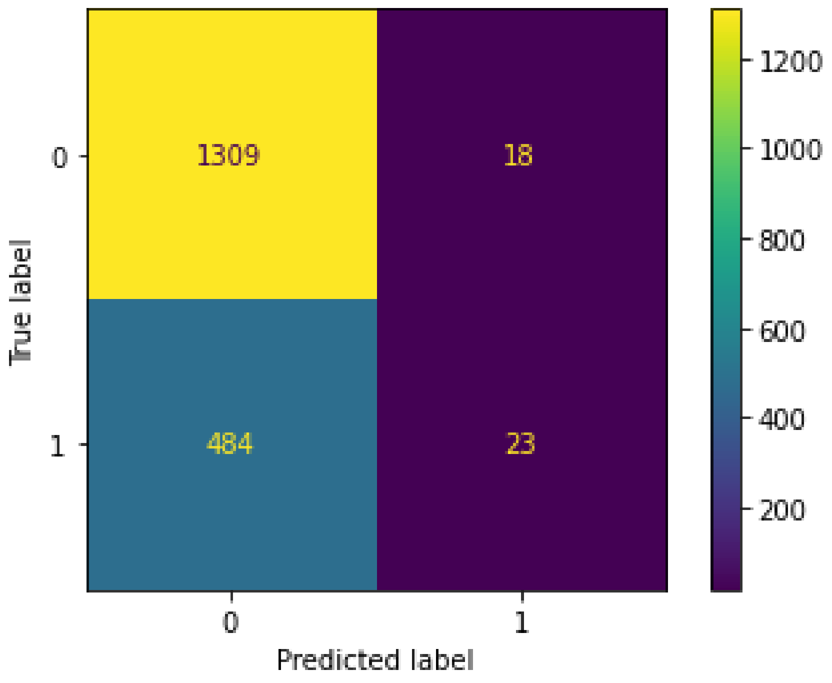
```
[45]: # Question 1 Code

      sephoraNew = pd.get_dummies(sephora, columns = ["category"])
      sephoraNew.head()

      predictors = ["number_of_reviews", "love", "price", "rating", "category_Spa␣
       ↪Tools", "category_Sponges & Applicators", "category_Sunscreen",␣
       ↪"category_Teeth Whitening", "category_Tinted Moisturizer",␣
       ↪"category_Toners", "category_Tweezers & Eyebrow Tools", "category_Value &␣
       ↪Gift Sets", "category_Wellness", "category_no category"]

      X = sephoraNew[predictors]
      y = sephoraNew["exclusive"]

      X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)

      z = StandardScaler()

      contin = ["love","rating","number_of_reviews","price"]

      z.fit(X_train[contin])
      X_train[contin] = z.transform(X_train[contin])
      X_test[contin] = z.transform(X_test[contin])
```

```
lr = LogisticRegression()
lr.fit(X_train[predictors], y_train)

#confusion matrix
print("The order of visuals: ")
print("Train Confusion Matrix")
plot_confusion_matrix(lr, X_train[predictors], y_train)

print("Test Confusion Matrix")
plot_confusion_matrix(lr, X_test[predictors], y_test)

#ROC/AUC curve
print("Train ROC curve")
plot_roc_curve(lr, X_train[predictors], y_train)

print("Test ROC curve")
plot_roc_curve(lr, X_test[predictors], y_test)
```

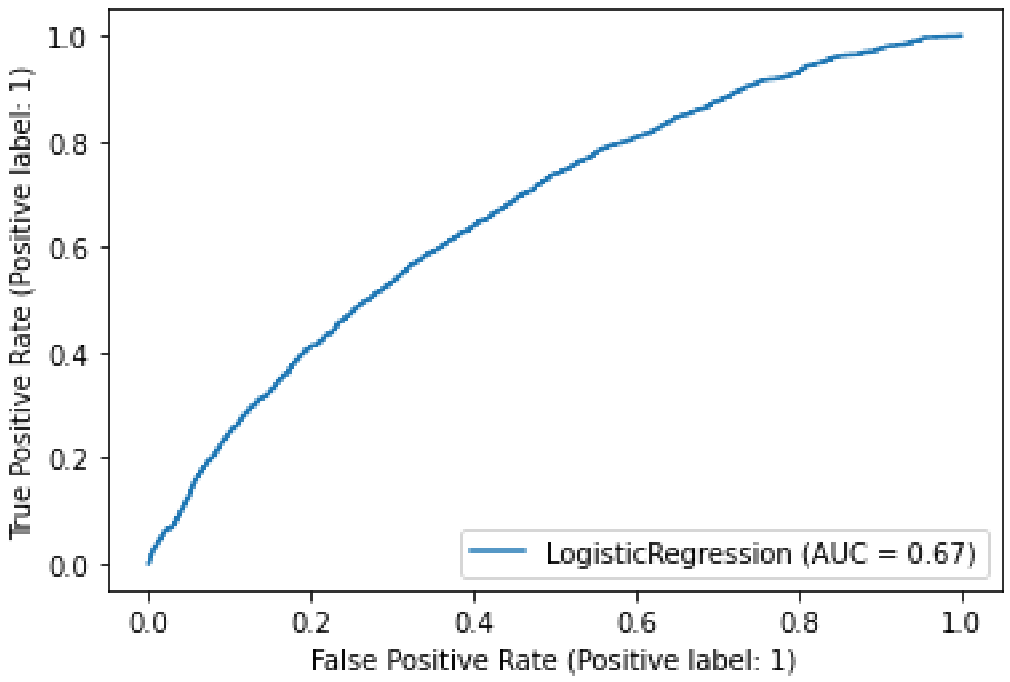
```
The order of visuals:
Train Confusion Matrix
Test Confusion Matrix
Train ROC curve
Test ROC curve
```

```
[45]: <sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x7fbd78b2d370>
```

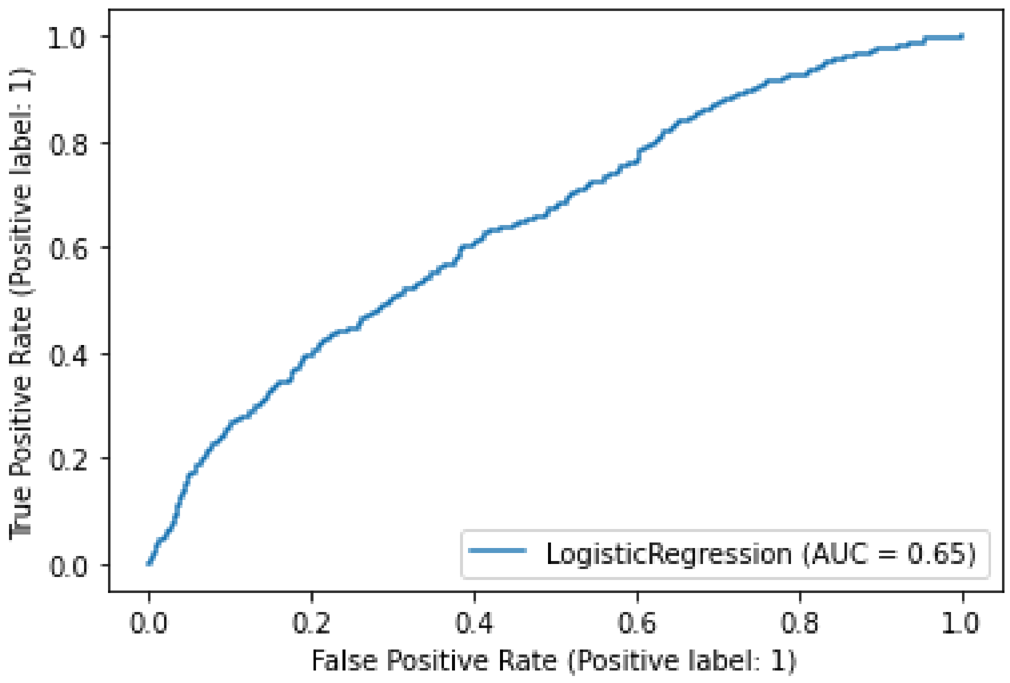




True label
Predicted label
0
1
1309
18
484
23
1200
1000
800
600
400
200



True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
LogisticRegression (AUC = 0.67)
0.0
0.2
0.4
0.6
0.8
1.0

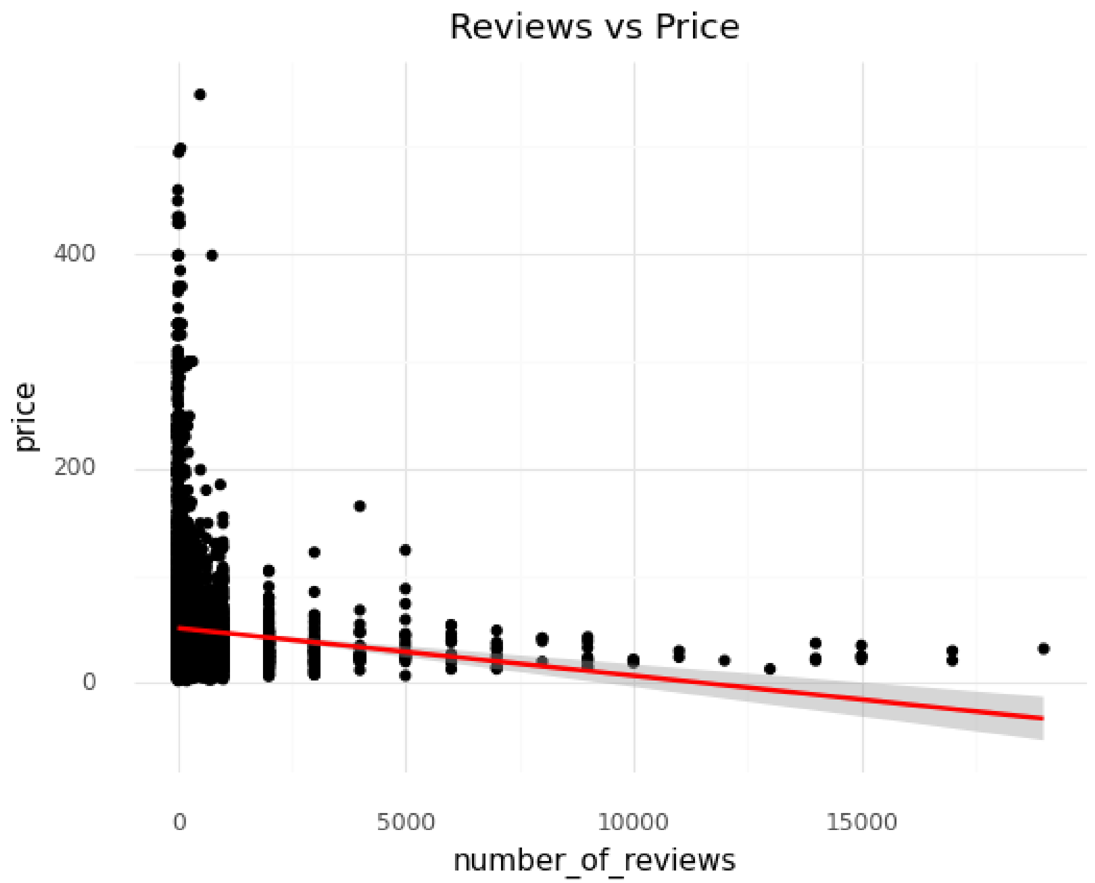


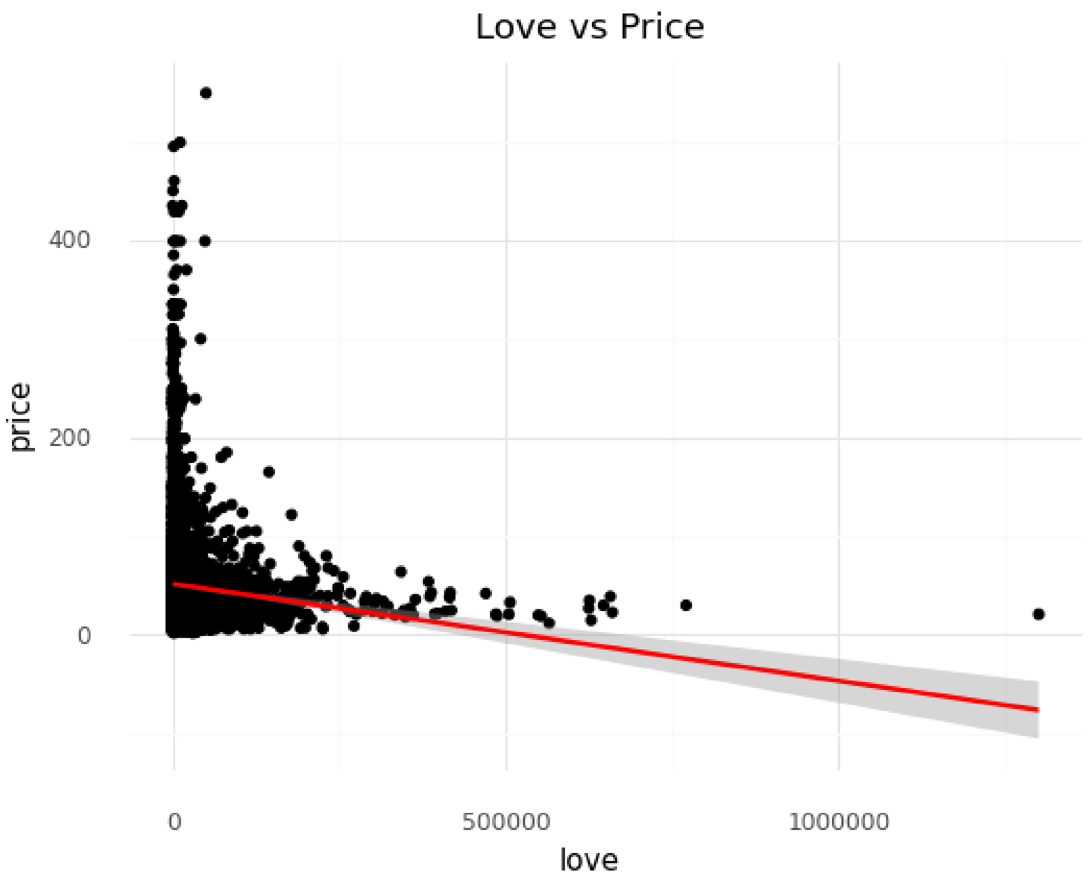
1.3 Question 2 - Sreya

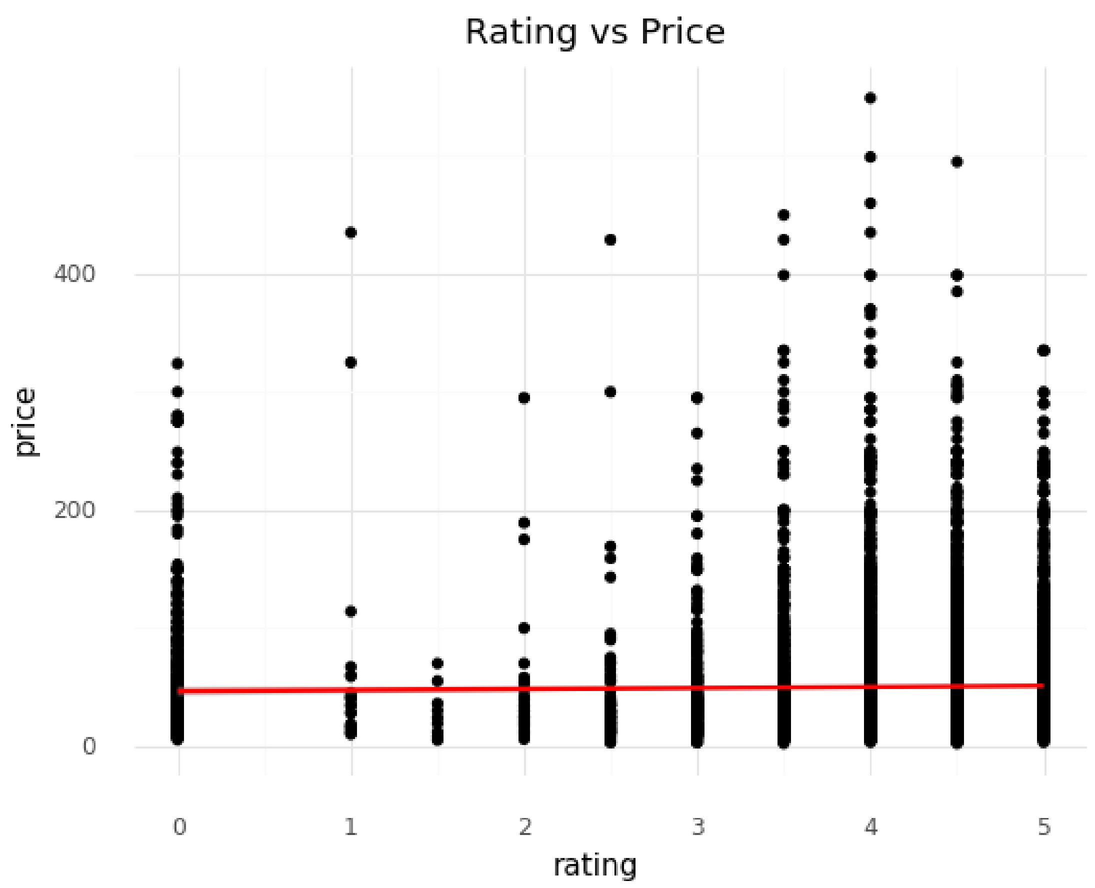
While predicting the price of a Sephora products, out of the predictors consisting of rating, number of reviews, love, and dummy variables of each category, which predictor has the most significant relationship with the price, either negative or positive?(linear regression)

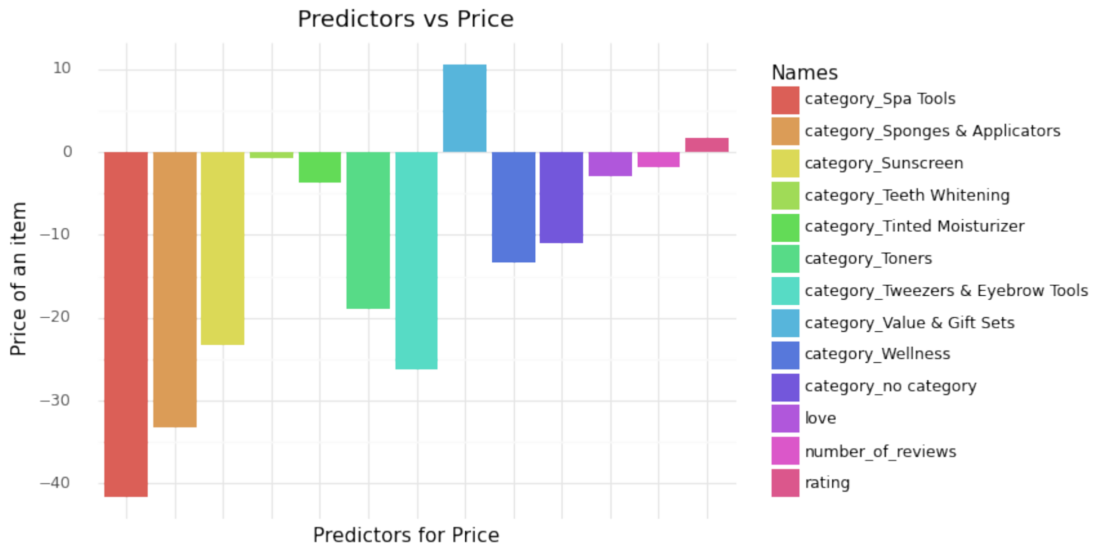
[46]:
```
#Question 2 Code

predictors = ["number_of_reviews", "love", "rating", "category_Spa Tools",␣
↪"category_Sponges & Applicators", "category_Sunscreen", "category_Teeth␣
↪Whitening", "category_Tinted Moisturizer", "category_Toners",␣
↪"category_Tweezers & Eyebrow Tools", "category_Value & Gift Sets",␣
↪"category_Wellness", "category_no category"]
contin = ["number_of_reviews", "love", "rating"]

X_train, X_test, y_train, y_test = train_test_split(sephoraNew[predictors],␣
↪sephoraNew["price"], test_size = .1)

z = StandardScaler()
z.fit(X_train[contin])
X_train[contin] = z.transform(X_train[contin])
X_test[contin] = z.transform(X_test[contin])

lr = LinearRegression()
```

```
lr.fit(X_train[predictors], y_train)

# Reviews vs Price
print(ggplot(sephoraNew, aes(x="number_of_reviews", y="price"))+ geom_point() +␣
 ↪ggtitle("Reviews vs Price")+theme_minimal() + geom_smooth(method = "lm",␣
 ↪color = "red"))

# Love vs Price
print(ggplot(sephoraNew, aes(x="love", y="price"))+ geom_point() +␣
 ↪ggtitle("Love vs Price")+theme_minimal()+geom_smooth(method = "lm", color =␣
 ↪"red"))

# Rating vs Price
print(ggplot(sephoraNew, aes(x="rating", y="price"))+ geom_point() +␣
 ↪ggtitle("Rating vs Price")+theme_minimal()+geom_smooth(method = "lm", color␣
 ↪= "red"))

coefficients = pd.DataFrame({"Coef": lr.coef_, "Names": predictors})

# Coefficients Bar Graph
(ggplot(coefficients, aes(x="Names", y="Coef", fill = "Names")) + labs(x =␣
 ↪"Predictors for Price", y = "Price of an item") + ggtitle("Predictors vs␣
 ↪Price") + geom_bar(stat = "identity") + theme_minimal() +
 theme(axis_text_x=element_blank()))
```


Reviews vs Price
price
400
200
0
0
5000
10000
15000
number_of_reviews


Love vs Price
price
400
200
0
0
500000
1000000
love


Rating vs Price
price
400
200
0
0
1
2
3
4
5
rating


Predictors vs Price
10
0
−10
−20
−30
−40
Price of an item
Predictors for Price
Names
category_Spa Tools
category_Sponges & Applicators
category_Sunscreen
category_Teeth Whitening
category_Tinted Moisturizer
category_Toners
category_Tweezers & Eyebrow Tools
category_Value & Gift Sets
category_Wellness
category_no category
love
number_of_reviews
rating

```
[46]: <ggplot: (8778230954719)>
```

Questions Response:

1) Using logistic regression, create a model in order to predict whether a Sephora product is exclusive or not based on the predictors rating, number of reviews, love, category, and price. Create a confusion matrix based on the results, and how does the train look in comparison to the test?

The first question that I worked on was a logistic regression question where I created a model in order to figure out whether or not a product was exclusive based on the variables number_of_reviews, love, price which were continuous and category of the product which was categorical. Therefore, I created a dummy variable for the category variable and added the dummy variables to the predictors along with the continuous variables. Then, I z-scored the continuous variables, and made the logistic regression model after doing an 80/20 train/test split. For the graphs, I decided to make confusion matrices and plot ROC/AUC curves for the train/test sets as well. This is what I saw when I looked at the results. First of all, I made confusion matrices, and these help determine the accuracy of a model when correctly predicting the predicted positives and negatives. The first confusion matrix describes the training set, and the top left corner represents the true negatives where the model predicted a negative, or the product was not exclusive, and this was actually correct. The amount of true negatives were in the 5,000s which is significantly high, and shows the model did very well in getting true negatives. Next, the top right corner was false positives where the model thought something was exclusive but it actually was not, and it did this about 70 or more times. The bottom left refers to a false negative where the model thought something was negative or not exclusive, but it was, and the times it did this was in the high thousands, and the true positives where it was actually exclusive, and the number of times the model predicted this was done was in the eighties. From these results, we can see that the model was much better at predicting true negatives more than true positives, so the model was much better at predicting if a product was not exclusive. In regards to the test set confusion matrix, the numbers were pretty similar, just smaller since it was a 80/20 split where the true negatives were predicted about 1,300 times and true positives were in the 20s so it was significantly less, so it was very similar to the training set showing that the model is not overfit. The next result I received was the ROC/AUC curve which shows the area under the curve, and the model is significantly better if the score for ROC is closer to 1 rather than to 0. The score for the training set was around the same value as the testing set where both were at the higher end of .6 closer to .7. This shows that this model could be better, but it is average since it is closer to 1 than it is to 0. Furthermore, since the values for the train and test sets are similar, this once again shows that the model is not overfit. Overall, the model was average at predicting whether or not a product was exclusive or not, and was much better at predicting true negatives.

2) While predicting the price of a Sephora products, out of the four predictors(use only these four to build a model) consisting of rating, number of reviews, love, and category, which predictor has the most significant relationship with the price, either negative or positive?(linear regression)

The second question that I worked on was a linear regression model that looked at which predictor between the predictors love, number_of_reviews, and rating which are all continuous and category, has the most significant effect/impact on the outcome of price. I also added in the dummy category variables as predictors, which is a change from the original question so that there would be more variables when making the PCA model. I z-scored all the continuous predictors and made a linear regression model, and then made 3 ggplots to see the effect of the continuous variables on price. I also made a 90/10 split, and fit the model on the training set, and testing would be for future data. Then I also added a trendline to the ggplots to make the trend patterns more clear.

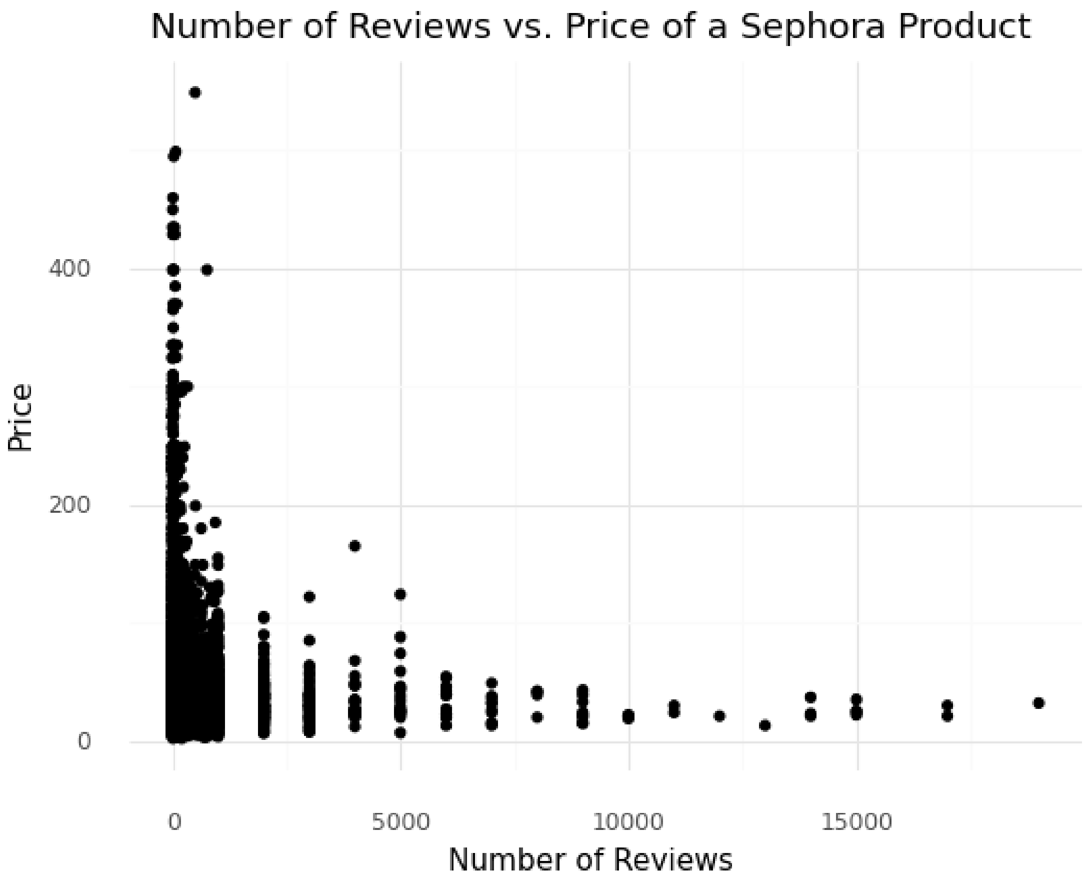
In regards to the ggplots, the first one was number_of_reviews on the x axis and price on the y axis, and it can be seen there in a negative trendline where the less number_of_reviews there were, the higher the price of the product was, and the same went for love, where the less people loved the product, the higher the price was, but I would assume the opposite where I would think that if a product was popular, it would be higher in price.The trend goes downward where as both these x variables increase, the price decreases and vice versa. There was a difference for the ggplot that measured rating and price. The final graph was rating vs price where the trend line does not show an evident trend, but at rating around 4, the price was pretty high. The trendline also is straight with no trend. After, I made a bar graph of the coefficients including the dummy category variables to see which had the most significant effect on price using the linear model that I created, and out of the continuous variables, love had the biggest negative impact and number_of_reviews also had a negative impact, but not as big, and rating may have had a small positive effect which is shown here, but not big enough to be seen in the scatterplot. Overall, to answer the question, love had the biggest impact negatively on price out of the continuous variables. In regards to all the variables including dummy variables for category, the category of spa_tools had the biggest negative impact on price.

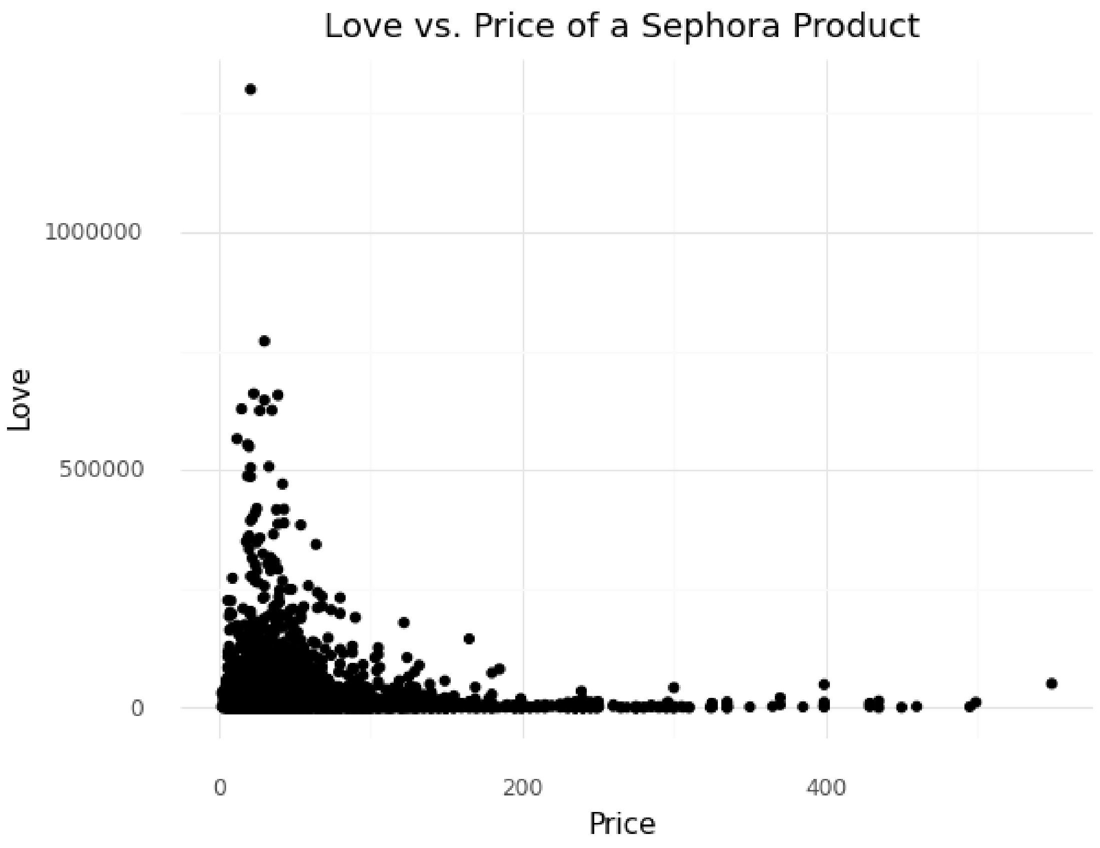
1.4 Question 3 - Sarah

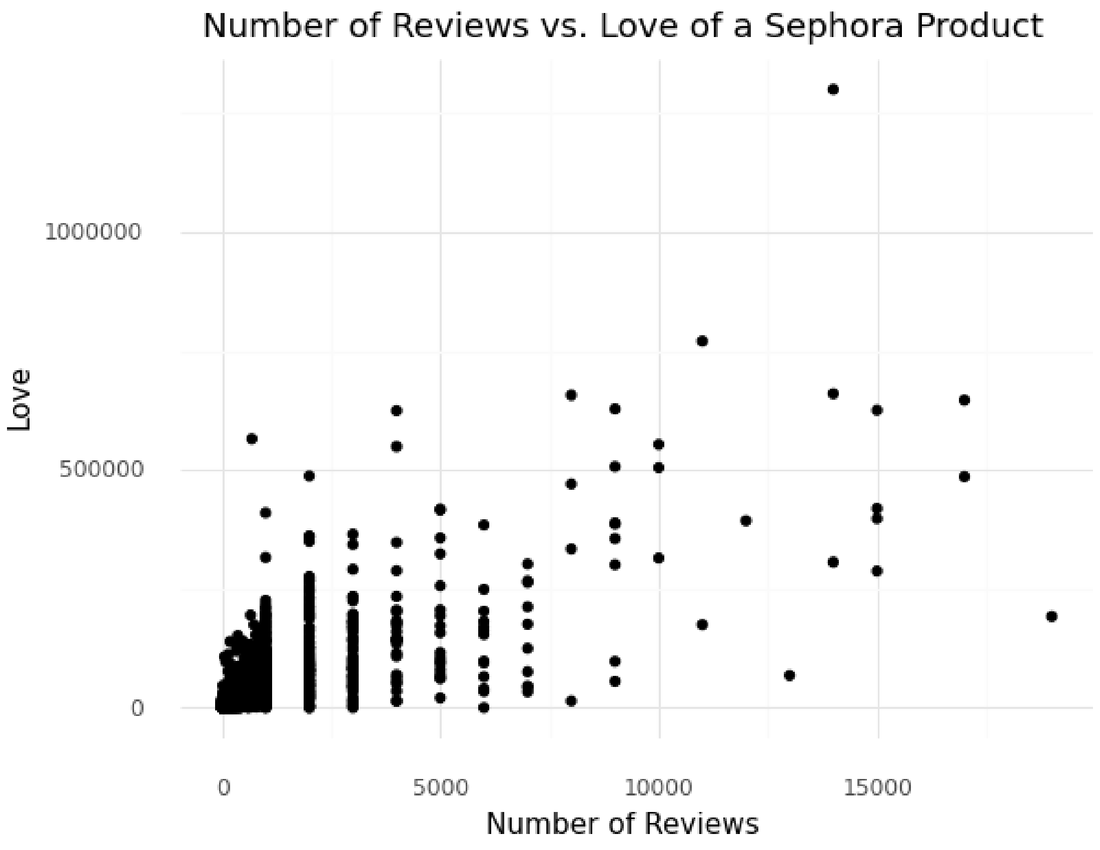
Between the different types of cluster models, would K-Means, DBSCAN, or EM Gaussian Mixture perform the best on clustering the products by number_of_reviews, love, and price? How would the silhouette scores compare to each other?

1.4.1 Data Exploration

[47]:
```
print((ggplot(sephora, aes(x = "number_of_reviews", y = "price")) +
        geom_point() +
        labs(title = "Number of Reviews vs. Price of a Sephora Product") +
        xlab("Number of Reviews") +
        ylab("Price") +
        theme_minimal()))

print((ggplot(sephora, aes(x = "price", y = "love")) +
        geom_point() +
        xlab("Price") +
        ylab("Love") +
        labs(title = "Love vs. Price of a Sephora Product") +
        theme_minimal()))

print((ggplot(sephora, aes(x = "number_of_reviews", y = "love")) +
        geom_point() +
        xlab("Number of Reviews") +
        ylab("Love") +
        labs(title = "Number of Reviews vs. Love of a Sephora Product") +
        theme_minimal()))
```




Love vs. Price of a Sephora Product
Love
1000000
500000
0
Price
0
200
400

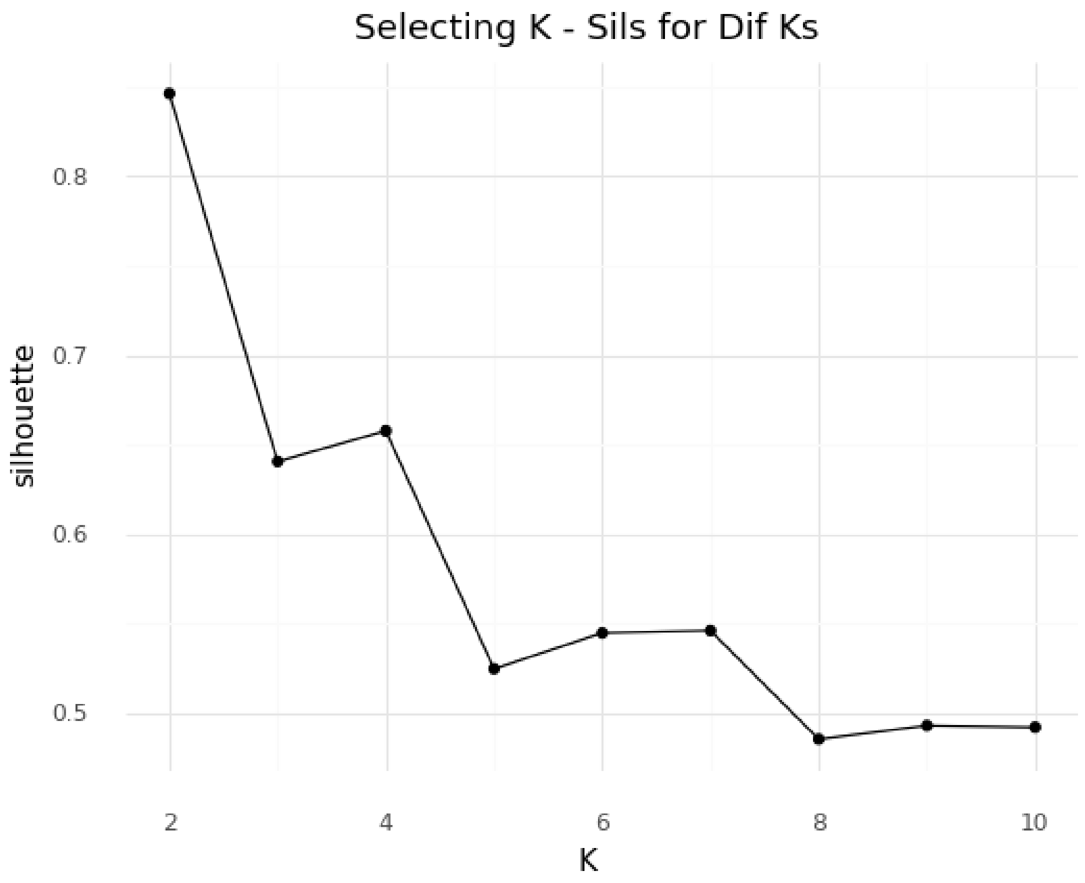


1.4.2 K Means Model

Selecting K K is the number of clusters permitted in a model. For EM Gaussian mixture and KMeans, we can use the silhouette scores of different K's to find the perfect K for our model.

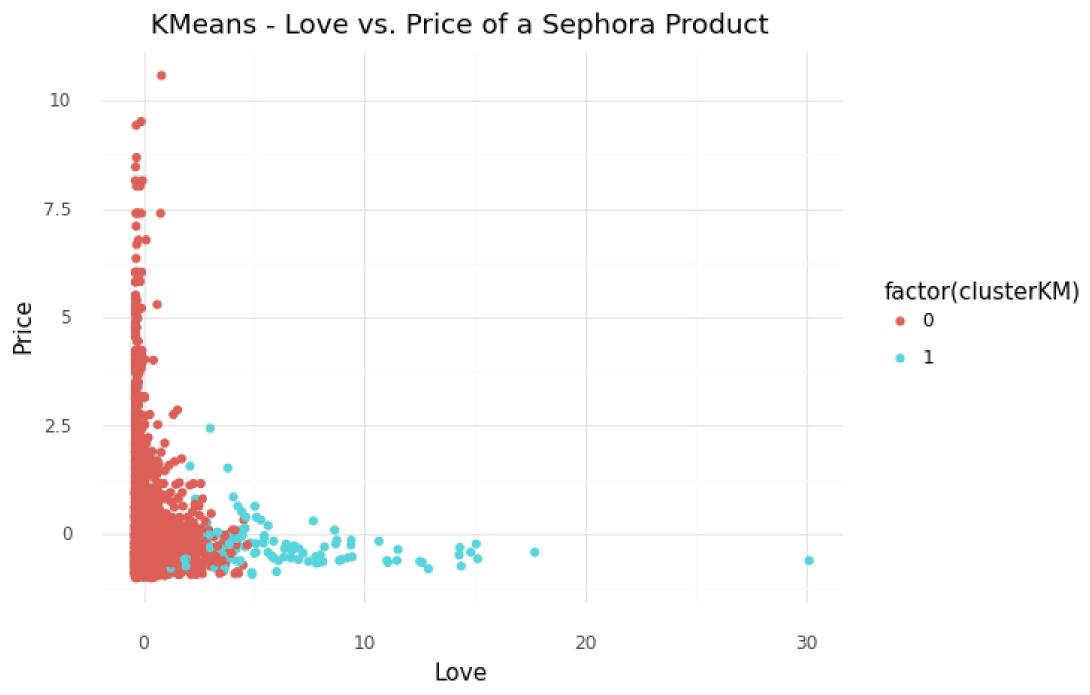
[48]:
```python
# Selecting K

variables = ["number_of_reviews","love","price"]
X = sephoraNew[variables]

z = StandardScaler()
X[variables] = z.fit_transform(X)

ks = [2,3,4,5,6,7,8,9,10]

sse = []
sils = []
```

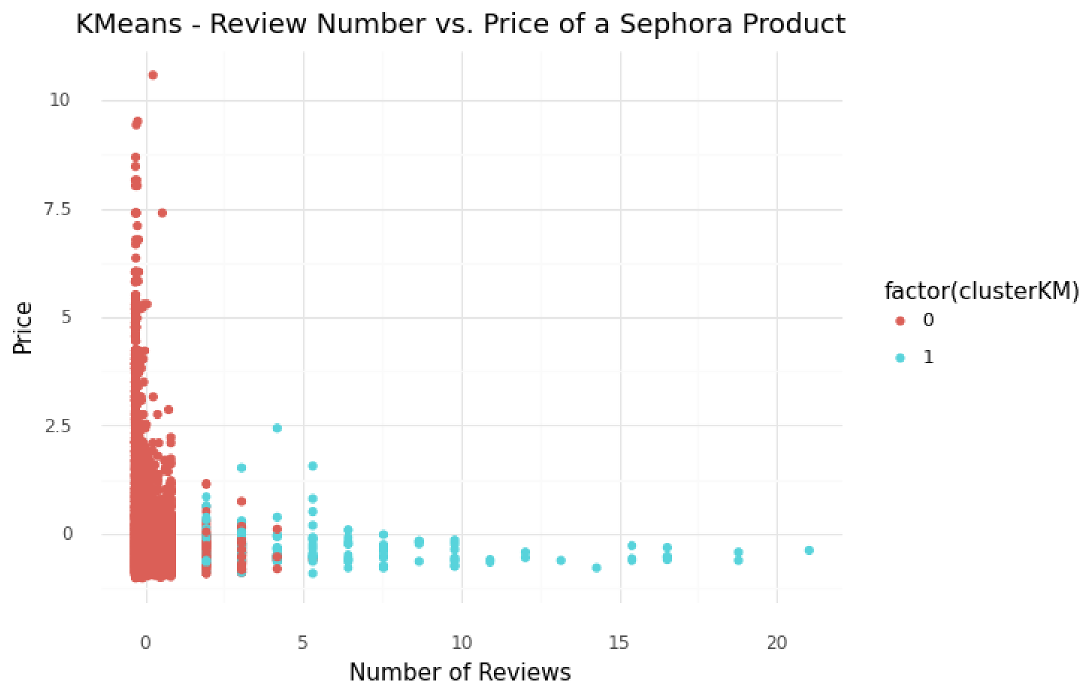
```
for k in ks:
    km = KMeans(n_clusters = k)
    km.fit(X)

    sse.append(km.inertia_)
    sils.append(silhouette_score(X, km.predict(X)))

sse_df = pd.DataFrame({"K": ks,
                       "sse": sse,
                       "silhouette": sils})

print((ggplot(sse_df, aes(x = "K", y = "silhouette")) + geom_point() +
 geom_line() +
 theme_minimal() +
 labs(title = "Selecting K - Sils for Dif Ks")))
```

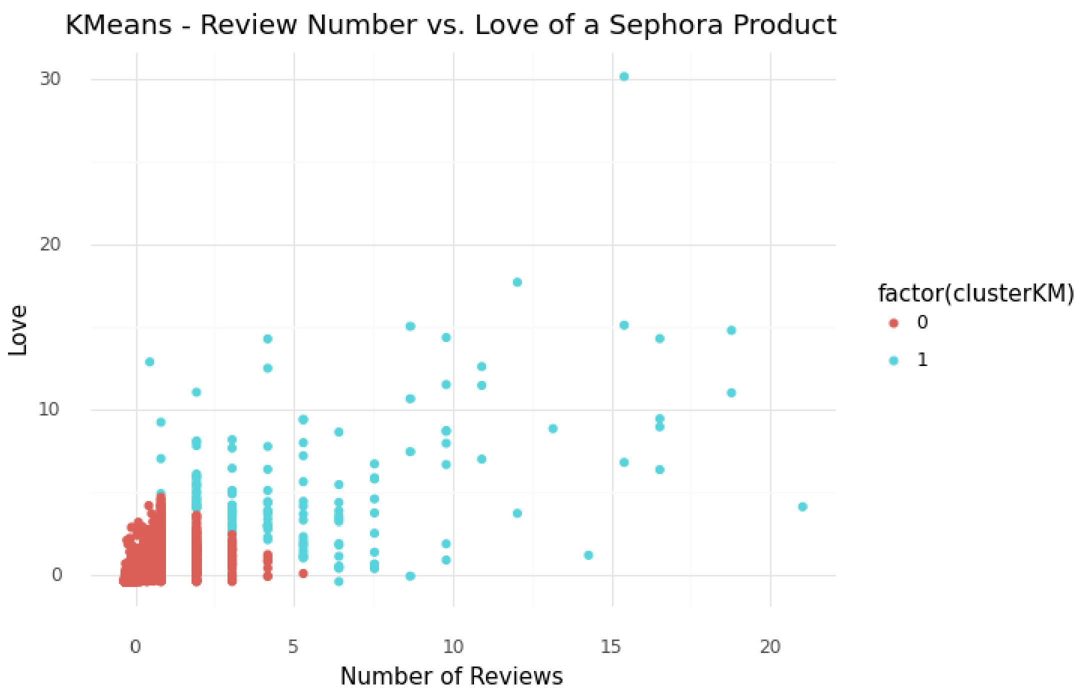


Building the KMeans Model using k neighbors of 2

[49]:
```
km = KMeans(n_clusters = 2)
km.fit(X[variables])

X["clusterKM"] = km.predict(X[variables])
X["clusterKM"]

print((ggplot(X, aes(x = "love", y = "price", color = "factor(clusterKM)")) +
       geom_point() +
       xlab("Love") +
       ylab("Price") +
       labs(title = "KMeans - Love vs. Price of a Sephora Product") +
       theme_minimal()))

print((ggplot(X, aes(x = "number_of_reviews", y = "price", color =␣
 ↪"factor(clusterKM)")) +
       geom_point() +
       xlab("Number of Reviews") +
       ylab("Price") +
       labs(title = "KMeans - Review Number vs. Price of a Sephora Product") +
       theme_minimal()))

print((ggplot(X, aes(x = "number_of_reviews", y = "love", color =␣
 ↪"factor(clusterKM)")) +
       geom_point() +
       xlab("Number of Reviews") +
       ylab("Love") +
       labs(title = "KMeans - Review Number vs. Love of a Sephora Product") +
       theme_minimal()))
```


KMeans - Love vs. Price of a Sephora Product
Price
10
7.5
5
2.5
0
0
10
20
30
Love
factor(clusterKM)
0
1

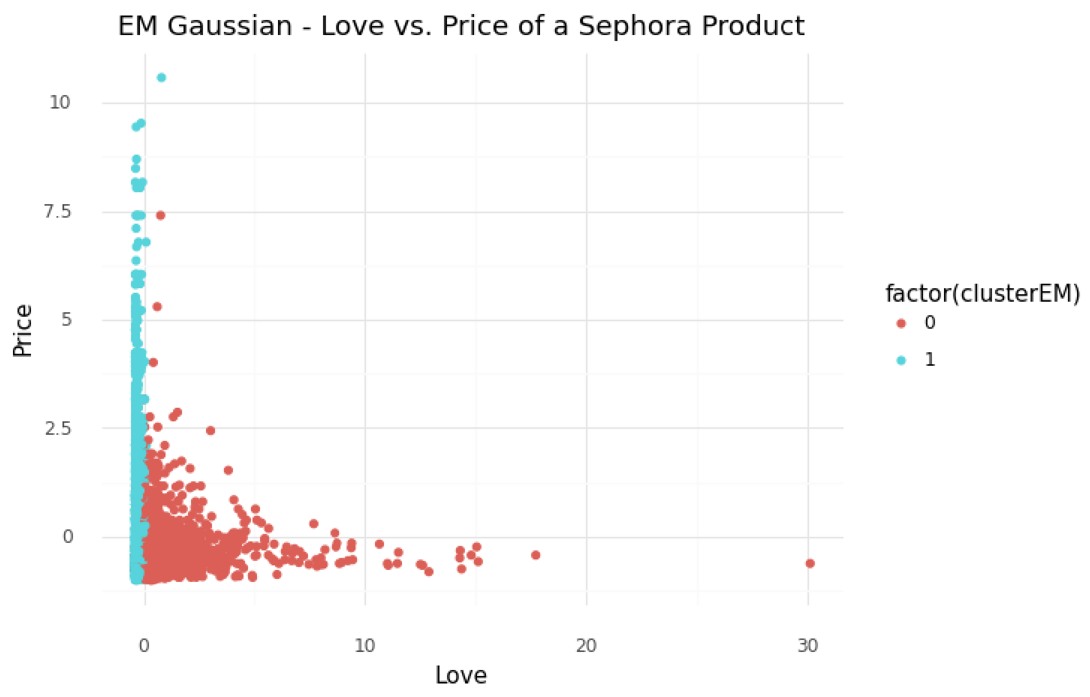

KMeans - Review Number vs. Price of a Sephora Product
Price
10
7.5
5
2.5
0
0
5
10
15
20
Number of Reviews
factor(clusterKM)
0
1

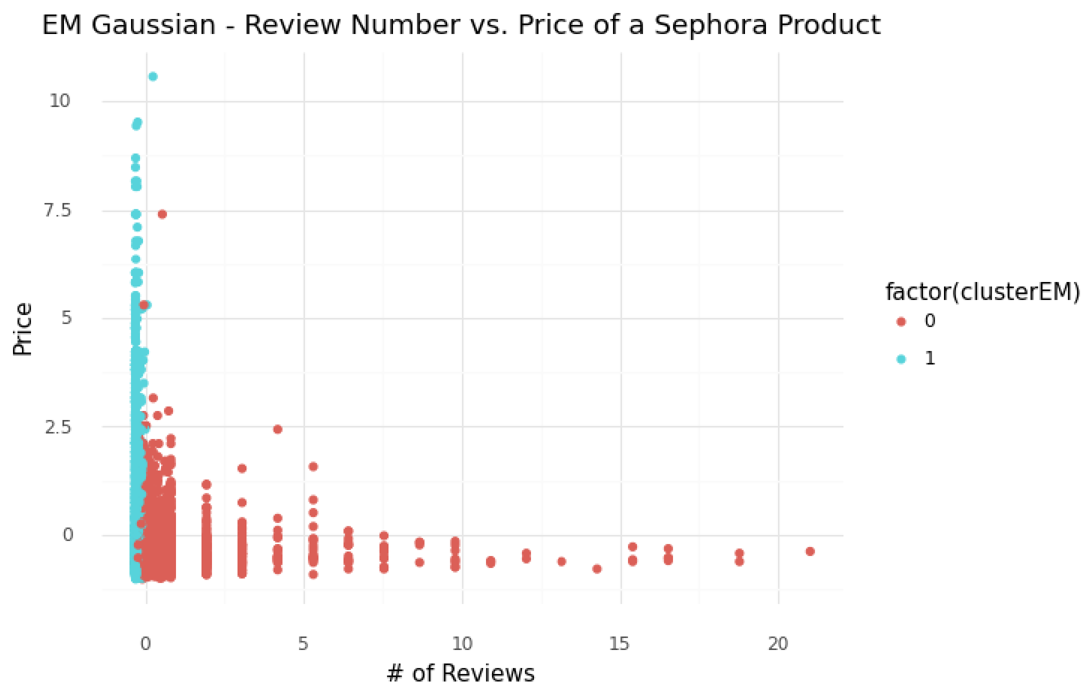


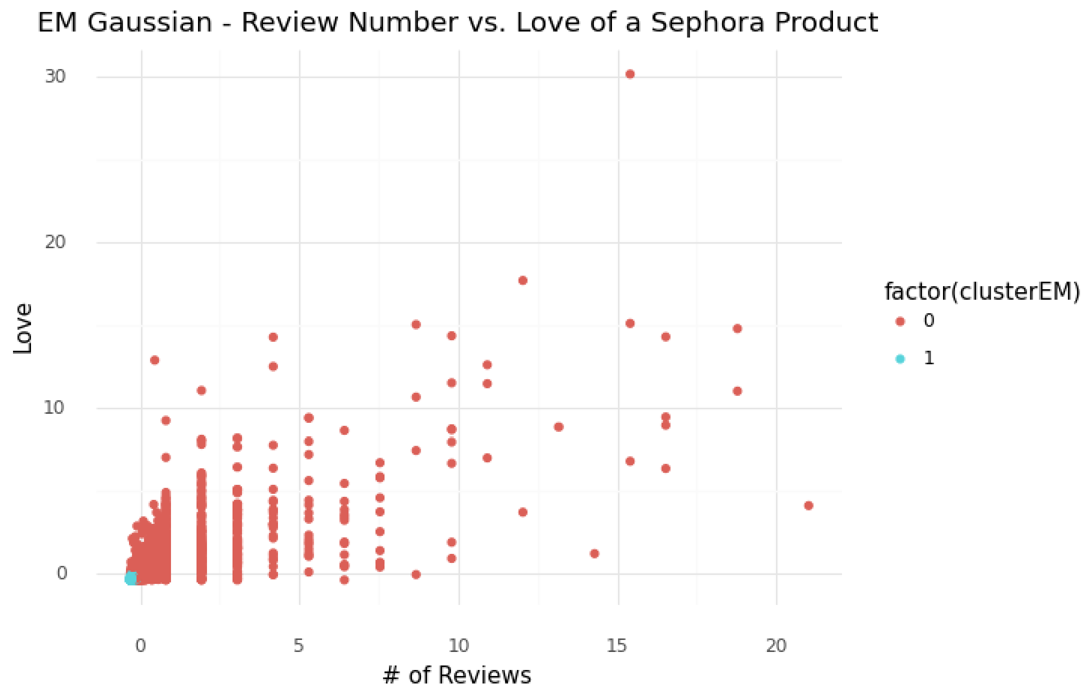
1.4.3 EM Gaussian Mixture Model

Building the model using k neighbors of 2

[50]:
```
# EM Gaussian Mixture
em = GaussianMixture(n_components = 2)
em.fit(X[variables])

X["clusterEM"] = em.predict(X[variables])

print((ggplot(X, aes(x = "love", y = "price", color = "factor(clusterEM)")) +
       geom_point() +
       xlab("Love") +
       ylab("Price") +
       labs(title = "EM Gaussian - Love vs. Price of a Sephora Product") +
       theme_minimal()))

print((ggplot(X, aes(x = "number_of_reviews", y = "price", color =␣
 ↪"factor(clusterEM)")) +
       geom_point() +
       xlab("# of Reviews") +
       ylab("Price") +
       labs(title = "EM Gaussian - Review Number vs. Price of a Sephora␣
↪Product") +
       theme_minimal()))

print((ggplot(X, aes(x = "number_of_reviews", y = "love", color =␣
↪"factor(clusterEM)")) +
       geom_point() +
       xlab("# of Reviews") +
       ylab("Love") +
       labs(title = "EM Gaussian - Review Number vs. Love of a Sephora␣
↪Product") +
       theme_minimal()))
```




EM Gaussian - Review Number vs. Price of a Sephora Product
Price
# of Reviews
factor(clusterEM)
0
1
10
7.5
5
2.5
0
0
5
10
15
20

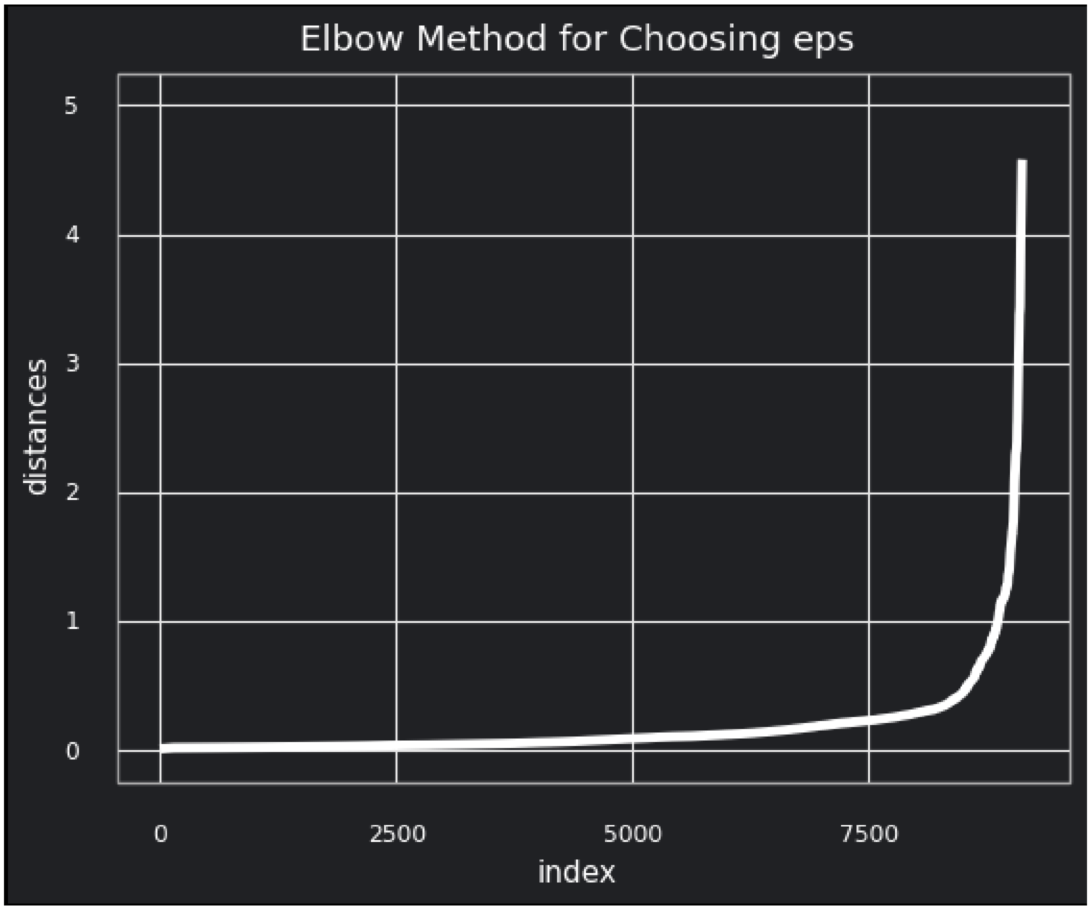

EM Gaussian - Review Number vs. Love of a Sephora Product
Love
# of Reviews
factor(clusterEM)
0
1
30
20
10
0
0
5
10
15
20

1.4.4 DBSCAN Model

DBSCAN is a different type of clustering model that can be used in a dataset that contains noise. It takes in three parameters: the distance metric, the epsilon/how far apart neighbors are, and the minimum points of each neighborhood. Since this is a dense dataset, I chose a higher minimum number of points, using 50. I also chose an epsilon of 0.5 based on the elbow curve.

Creating the model

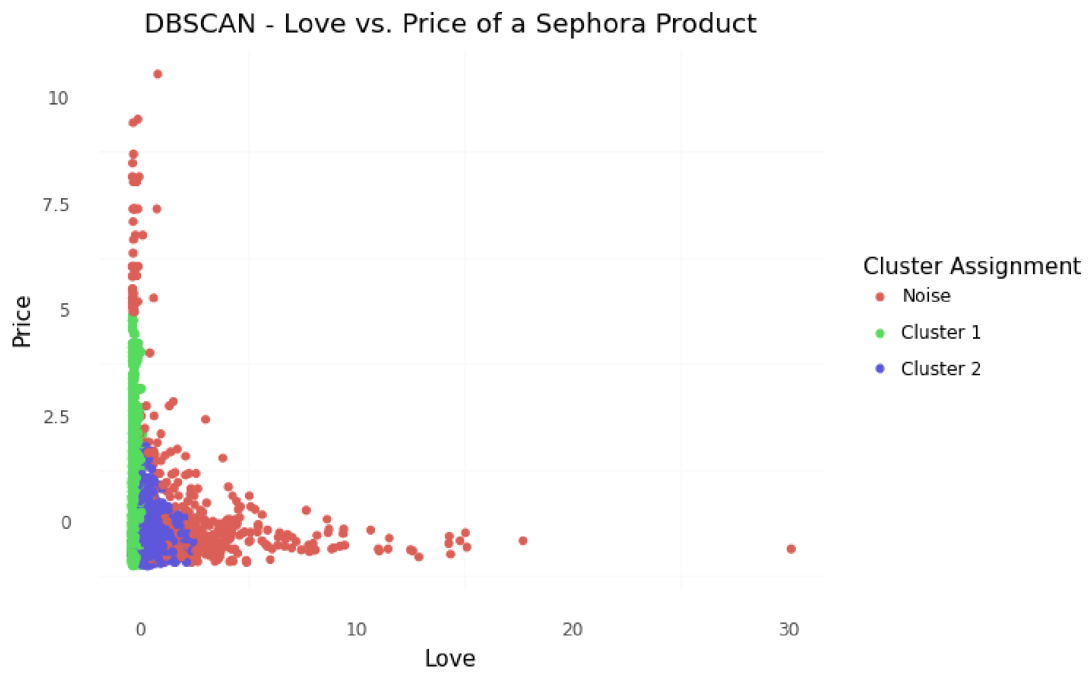
[51]:
```
# sort the distances
mins = 50
nn = NearestNeighbors(n_neighbors = mins + 1)

nn.fit(X[variables])

distances, neighbors = nn.kneighbors(X[variables])

distances = np.sort(distances[:, mins], axis = 0)

distances_df = pd.DataFrame({"distances": distances,
                             "index": list(range(0,len(distances)))})
plt = (ggplot(distances_df, aes(x = "index", y = "distances")) +
  ylim(0,5) +
 geom_line(color = "white", size = 2) + theme_minimal() +
 labs(title = "Elbow Method for Choosing eps") +
 theme(panel_grid_minor = element_blank(),
      rect = element_rect(fill = "#202124ff"),
      axis_text = element_text(color = "white"),
      axis_title = element_text(color = "white"),
      plot_title = element_text(color = "white"),
      panel_border = element_line(color = "darkgray"),
      plot_background = element_rect(fill = "#202124ff")
      ))
ggsave(plot=plt, filename='elbow.png', dpi=300)

plt

# Choosing 0.5 Eps!
```

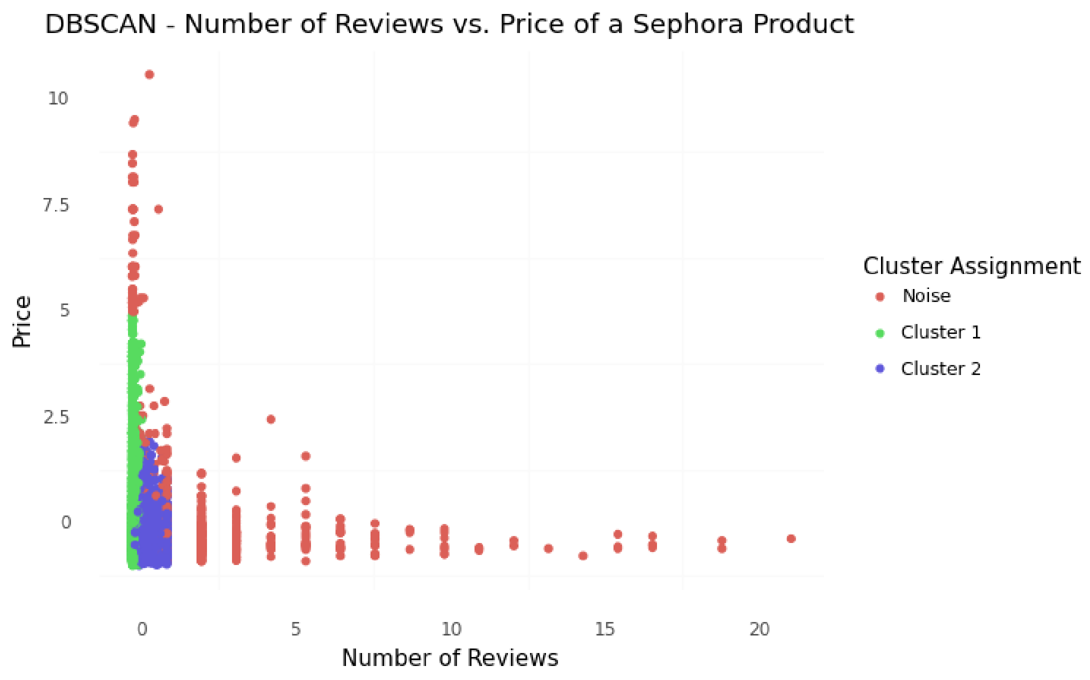


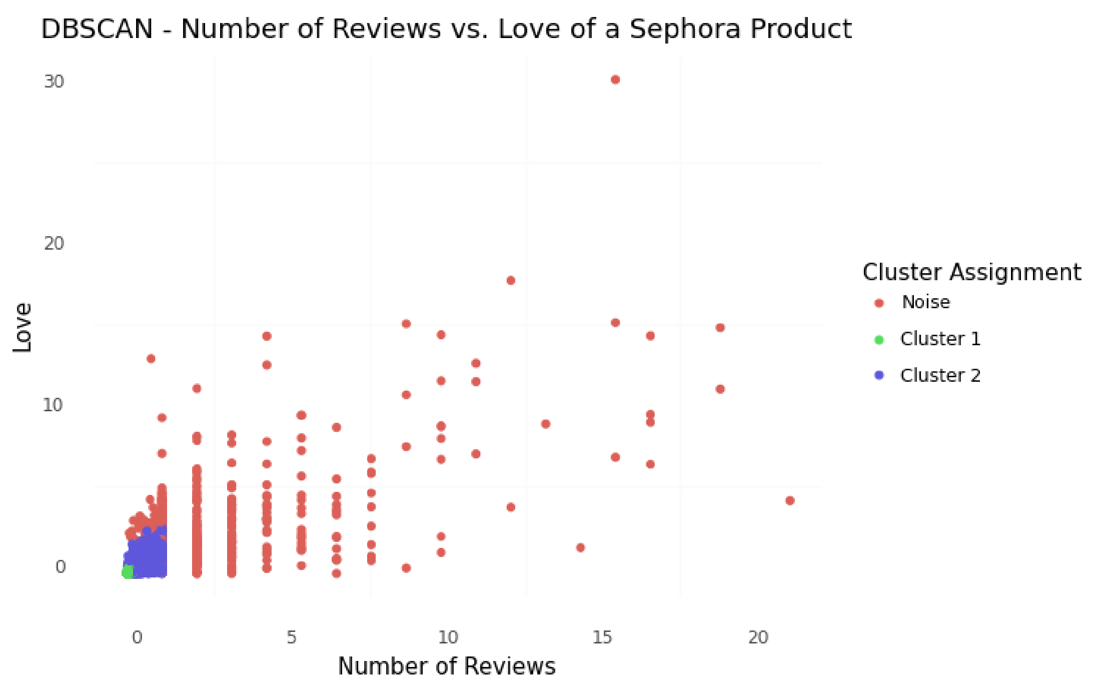
```
[51]: <ggplot: (8778224641015)>
```

DBSCAN Visuals Based on the visuals, we can tell that our three models perform in interestingly unique ways. This can be due to the assumptions made by the respective models. We will discuss this in the final answer below, along side the silhouette score metrics. These metrics will tell us which model had the most cohesive and separate clusters.

```
[52]: # DBSCAN Visual
db1 = DBSCAN(eps = 0.5, min_samples = 50).fit(X)

labsList = ["Noise"]
labsList = labsList  + ["Cluster " + str(i) for i in range(1,len(set(db1.
 ↪labels_)))]

X["clusterDBSCAN"] = db1.labels_

print((ggplot(X, aes(x = "love", y = "price", color = "factor(clusterDBSCAN)"))␣
 ↪+
 geom_point() +
 theme_minimal() +
 xlab("Love") +
  ylab("Price") +
 scale_color_discrete(name = "Cluster Assignment",
                      labels = labsList) +
theme(panel_grid_major = element_blank()) +
labs(title = "DBSCAN - Love vs. Price of a Sephora Product")))

print((ggplot(X, aes(x = "number_of_reviews", y = "price", color =␣
 ↪"factor(clusterDBSCAN)")) +
 geom_point() +
 theme_minimal() +
 xlab("Number of Reviews") +
 ylab("Price") +
 scale_color_discrete(name = "Cluster Assignment",
                      labels = labsList) +
theme(panel_grid_major = element_blank()) +
labs(title = "DBSCAN - Number of Reviews vs. Price of a Sephora Product")))

print((ggplot(X, aes(x = "number_of_reviews", y = "love", color =␣
 ↪"factor(clusterDBSCAN)")) +
 geom_point() +
 theme_minimal() +
 xlab("Number of Reviews") +
 ylab("Love") +
 scale_color_discrete(name = "Cluster Assignment",
                      labels = labsList) +
theme(panel_grid_major = element_blank()) +
labs(title = "DBSCAN - Number of Reviews vs. Love of a Sephora Product")))
```


DBSCAN - Love vs. Price of a Sephora Product
Price
Love
0
2.5
5
7.5
10
0
10
20
30
Cluster Assignment
Noise
Cluster 1
Cluster 2


DBSCAN - Number of Reviews vs. Price of a Sephora Product
Price
Number of Reviews
0
2.5
5
7.5
10
0
5
10
15
20
Cluster Assignment
Noise
Cluster 1
Cluster 2



1.4.5 Metrics

Measuring Silhouette Score of each model.

[53]:
```
print("Silhouette for KMeans ", silhouette_score(X[variables],X["clusterKM"]))
print("Silhouette for EM Gaussian Model ",␣
↪silhouette_score(X[variables],X["clusterEM"]))
print("Silhouette for DBSCAN Model ",␣
↪silhouette_score(X[variables],X["clusterDBSCAN"]))
```

```
Silhouette for KMeans  0.8460708503371711
Silhouette for EM Gaussian Model  0.2894412238760454
Silhouette for DBSCAN Model  0.1968922462032564
```

1.5 Question 3 Response

Between the different types of cluster models, would K-Means, DBSCAN, or EM Gaussian Mixture perform the best on clustering the products by number_of_reviews, love, and price? How would the silhouette scores compare to each other?

Initially, we were going to include Hierarchial Agglomerative Clustering, but since there's not really an intuitive hierarchy to the data, that made modeling difficult. We also wanted to use the "rating" variable, but since it either uses whole numbers or halves from 1-5, we felt this would be difficult to cluster with. We also just wanted it to look less categorized visually, so we swapped it for number_of_reviews, which is more of a continous variable.

For our clustering models, we selected KMeans, DBSCAN, and EM Gaussian Mixture models. These were selected because they all work to achieve a similar goal of clustering alike datapoints, or items at Sephora, together. However, these models have different assumptions of what these clusters could look like. KMeans is a simple and computationally efficient method of clustering that assumes spherical variance that has roughly the same number of data points in each cluster. DBSCAN assumes that there will be noise/outliers and oddly shaped clusters. EM Gaussian Mixture uses soft assignments, meaning that it assigns the probability of an item being in a cluster instead of a strict assignment. It also assumes different variances. This means that data points can be spread out in varied amounts, while still being a cluster. It assumes elliptical-shaped clusters.

When we explored the data in the first few visualization scatterplots, we gathered that there is not a lot of intuitive clustering occuring at first glance. It looks like the items have a lot of correlation with each other based on the variables we chose. For example, items with low prices had high love scores and high number of ratings, probabilty due to their inexpensive price making them more affordable for everyone. More sales can lead to more feedback from customers. Despite the shape of our data, we still wanted to see how these clustering models will perform.

Part of creating these models require selection of parameters, or certain arguments that make the model perform properly. For KMeans and EM Gaussian Mixture, we are required to provide the model with k, the number of clusters. We find k by looking at a plot that dispays the number of k that provides the highest silhouette score. A silhouette score is a metric that measures the separation and cohesion of clusters in the model. We do this by making a graph that plots the silhouette scores at each number of k. The k with the highest silhouette score ended up being 2.

However, DBSCAN uses a different type of parameters, requiring distance metrics, epsilons, and minimum points to be included in the model. To obtain these parameters, we must do some separate hyperparameter tuning, unique to DBSCAN. There are two parameters we must choose: minimum points and epsilon, or how far from a point are neighbors. We can select minimum points, or the minimum amount of density to have a cluster, based on domain knowledge and how the dataset looks. It is wise to select a higher minimum points if they have more rows, noise, or features. I picked a higher minimium point, 50. To select epsilon, we can use domain knowledge as well, but in this demonstration we use the ”elbow method.” This is a method that allows us to choose epsilon based on a graph that compares distances and indexes. This is also known as a k-dist graph, which shows that for every datapoint in the dataset, we will say how far away is it’s k-th nearest neighbor. We order the distances to make this graph, and from it we get a plot that shows the relationship. We then look for the point of inflection on the plot, and from that, we can select our epsilon. For this model, we used an epsilon of 0.5.

After each model was created, visualizations of scatterplots were made to display the relationships between each cluster assignments. For KMeans, I made visuals for # of Reviews and Price. Based on scatterplot, it looks like one cluster is really condensed towards the y axis and origin, and another is more concentrated towards the x axis and farther away from the origin. I made another scatterplot showing love versus price with clustering, and it looks like there’s a cluster that has low love scores, and a cluster that has high love scores. The price between the low love scores cluster is pretty varied, but the high cluster stays pretty concentrated to low prices. Then, I made a scatterplot showing of Reviews vs Love scores. This graph shows one cluster that has products with low user feedback interaction, and one with high feedback interaction. This means high love scores and high number of reviews.

I made the same graphs including the EM Gaussian Mixture assignments. For the graph that shows love and reviews, we can see there is a major dominance in the blue cluster, with little presence of the other cluster. This is not very intuitive for viewing, as the major cluster includes items with high and low numbers of reviews and love scores. This could be due to EM's assumption of diverse variance in clustering. Love versus price and number of reviews versus price show similar clustering patterns, where the red clusters both seem concentrated to the y axis, showing little to no units of love or number of reviews. The blue clusters are more spread out, and feature higher units love and numbers of reviews respectively.

Lastly, I made visuals to explore the clustering assignments for DBSCAN. From the visuals, I noticed that there was a major presence of nise, or outliers in this model. The actual clustering occurs closer to the origin of the graph in all three of these visuals, which is also not intuitive on the eyes.

Once we explored how the clusters interact visually, we continued the model exploration by featuring metrics of the silhouette scores per model. Based on the Silhouette Scores seen for each model, we can gather that the KMeans clustering model performed the best. A silhouette score is a metric that measures the separation and cohesion of clusters in the model. Having a higher unit as the silhouette score is favorable, as it means our model is producing more separate, cohesive, and therefore distinct clusters. By having more distinct clusters, we can differentiate them and witness trends between different types of Sephora products.

The KMeans model has a score of 0.846, which is the closest to 1. Our EM Gaussian Mixture Model follows, with a silhouette score of 0.289. Lastly, our DBSCAN model received a score of about 0.188. If we look at the visuals of KMeans, we see that cluster one and two share a lot of space on the graph. They are not evenly split, but there is more of cluster 2 in KMeans than in EM Gaussian Mixture. In EM Gaussian Mixture, there was a heavy reliance on cluster 1. I believe this is because Gaussian Mixture assumes that data points can be spread out in varied amounts and still be a cluster with an elliptical shape. KMeans, on the other hand, clusters more uniformily. Lastly, DBSCAN performed so poorly because our data contains clusters that do have different densities. The less dense clusters might have been mistaken for noise.

Based on the visuals, we can gather that there are some interesting data clustering occuring within each model. Personally, after investigating the behavior of these models with these predictors, I am not sure if I would use clustering, due to it's unseemly visuals and unintuitive groupings. Maybe we would have to hone in on different ranges of data to recognize more patterns, and do some deeper data cleaning. However, based on the metrics and the best looking visual, I would answer this question by saying **KMeans is the best model when examining the clusters that occur in relation to number_of_reviews, love, and price.**

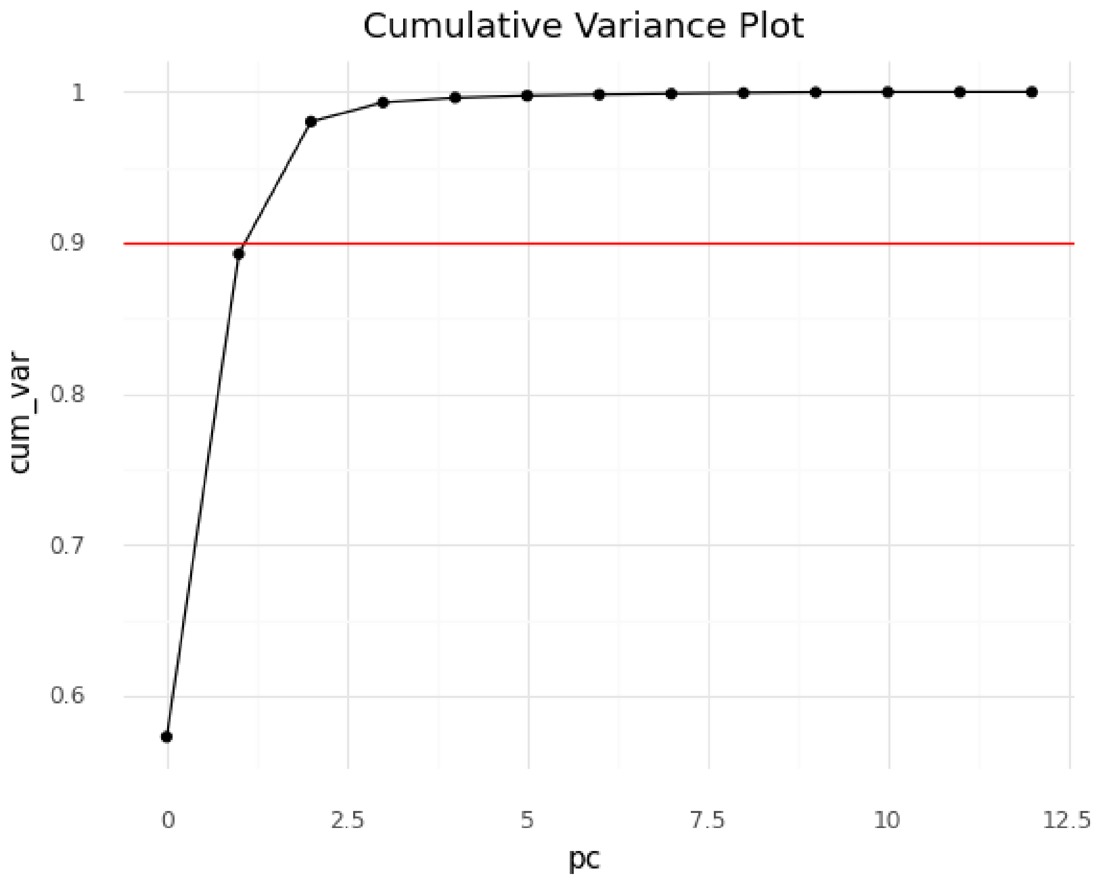
1.6 Question 4 - Sarah

Dimensionality Reduction - How would PCA compare with Regular Linear Regression on the same formula from the second hypothesis? To retain 90% of original variance How would their metrics compare? Based on these scores, which model would have better error and less overfitting, making it more suited to test with?

1.6.1 PCA Model Construction

Cumulative Variance Plot - Selecting the amount of Principal Components to keep for our PCA model.

[54]:
```
# Question 4 Code

# Trying PCA
pca = PCA()
pca.fit(X_train)

pcaDF = pd.DataFrame({"expl_var": pca.explained_variance_ratio_,
                     "pc": range(0,13), # num of components, second will change␣
 ↪based on num of variables
                     "cum_var": pca.explained_variance_ratio_.cumsum()}) #␣
 ↪grabs cumulative variance

(ggplot(pcaDF, aes(x = "pc", y = "cum_var")) +
 geom_line() +
 geom_point() +
 labs(title = "Cumulative Variance Plot") +
 geom_hline(yintercept = 0.9, color = "red")) + theme_minimal()
```

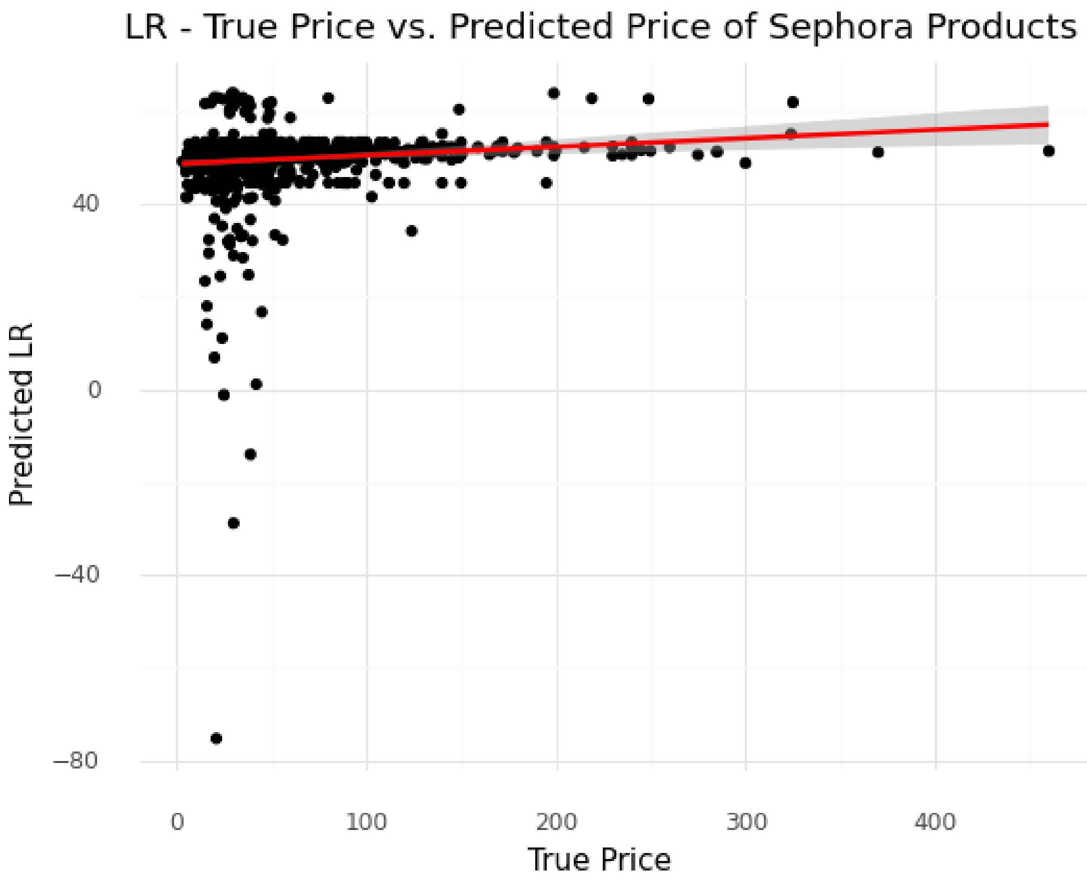


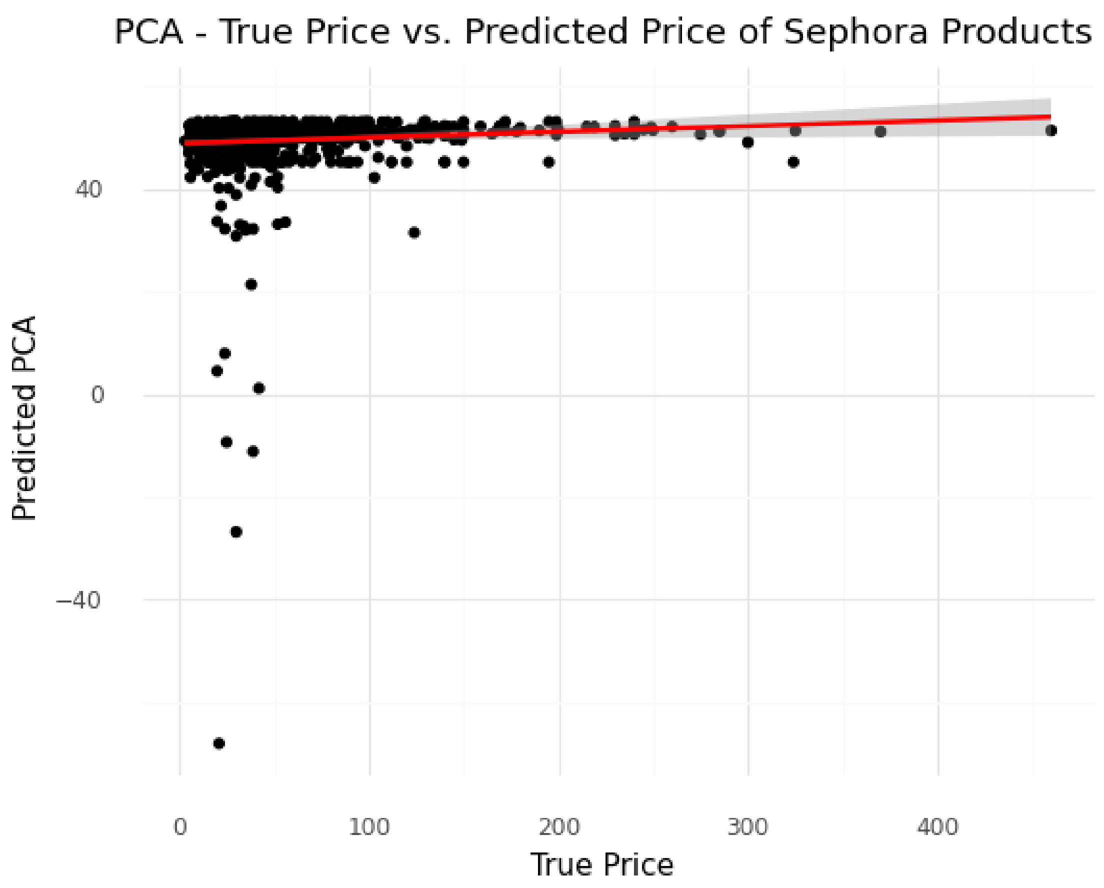
[54]:
```
<ggplot: (8778225481936)>
```

1.6.2 Building a new Linear Regression model with PCA

Exploring Accuracy with a Scatterplot. Thees two plots compares the predicted values with the true values.

[55]:
```
# 2 PC's
pcomps2_train = pca.transform(X_train)
pcomps2_train = pd.DataFrame(pcomps2_train[:,0:2])

pcomps2_test = pca.transform(X_test)
pcomps2_test = pd.DataFrame(pcomps2_test[:,0:2])

lr2 = LinearRegression()
lr2.fit(pcomps2_train, y_train)

vals = pd.DataFrame({"True Price": y_test,
                      "Predicted LR":lr.predict(X_test),
                      "Predicted PCA": lr2.predict(pcomps2_test)
                     })

print((ggplot(vals, aes(x = "True Price", y = "Predicted LR")) +
       geom_point() +
       labs(title = "LR - True Price vs. Predicted Price of Sephora Products") +
       geom_smooth(method = "lm", color = "red") +
       theme_minimal()))

print((ggplot(vals, aes(x = "True Price", y = "Predicted PCA")) +
       geom_point() +
       labs(title = "PCA - True Price vs. Predicted Price of Sephora Products")␣
 ↪+
       geom_smooth(method = "lm", color = "red") +
       theme_minimal()))
```


LR - True Price vs. Predicted Price of Sephora Products
Predicted LR
True Price
40
0
−40
−80
0
100
200
300
400



1.6.3 Model Metrics

Exploring how accurate our models are with MAE, MSE, & R2.

[56]:
```
# Metrics
# MAE, MSE, R2

# MAE Metrics
mae_lr_train = mean_absolute_error(y_train, lr.predict(X_train))
mae_lr_test = mean_absolute_error(y_test, lr.predict(X_test))

mae_pca_train = mean_absolute_error(y_train, lr2.predict(pcomps2_train))
mae_pca_test = mean_absolute_error(y_test, lr2.predict(pcomps2_test))

print("MAE Train for Linear: ", mae_lr_train)
print("MAE Test for Linear: ", mae_lr_test)
```

```python
print("MAE Train for PCA: ", mae_pca_train)
print("MAE Test for PCA: ", mae_pca_test)

print(" ")

# MSE Metrics
mse_linear_test = mean_squared_error(y_test, lr.predict(X_test))
mse_linear_train = mean_squared_error(y_train, lr.predict(X_train))

mse_pca_test = mean_squared_error(y_test, lr2.predict(pcomps2_test))
mse_pca_train = mean_squared_error(y_train, lr2.predict(pcomps2_train))

print("MSE Train for Linear: ", mse_linear_train)
print("MSE Test for Linear: ", mse_linear_test)

print("MSE Train for PCA: ", mse_pca_train)
print("MSE Test for PCA: ", mse_pca_test)

print(" ")

# R2 Metrics
r2_linear_train =  r2_score(y_train, lr.predict(X_train))
r2_linear_test = r2_score(y_test, lr.predict(X_test))

r2_pca_train = r2_score(y_train, lr2.predict(pcomps2_train))
r2_pca_test = r2_score(y_test, lr2.predict(pcomps2_test))

print("R2 Train for Linear: ", r2_linear_train)
print("R2 Test for Linear: ", r2_linear_test)

print("R2 Train for PCA: ", r2_pca_train)
print("R2 Test for PCA: ", r2_pca_test)
```

```
MAE Train for Linear:  30.13688328024365
MAE Test for Linear:  31.5946513002429
MAE Train for PCA:  30.296685505227572
MAE Test for PCA:  31.64441180982981

MSE Train for Linear:  2165.5729568851366
MSE Test for Linear:  2395.9533390160436
MSE Train for PCA:  2180.0294778443067
MSE Test for PCA:  2416.974851102111

R2 Train for Linear:  0.016649367850402497
R2 Test for Linear:  0.009044022638624027
R2 Train for PCA:  0.010084902322383016
R2 Test for PCA:  0.00034961581707138034
```

1.7 Question 4 Response

How would PCA compare with Regular Linear Regression on the same formula from the second hypothesis? To retain 90% of original variance How would their Mean Absolute Errors compare? Based on these scores, which model would have better error and less overfitting, making it more suited to test with?

Initially, we planned on using the LASSO regression model. However, we did not feel it was appropriate as LASSO pulls coefficients to zero. We felt we could get more interesting and interpretable results from PCA instead. Also, we felt that including MSE and R2 Scores instead of Mean Average Error only could give us a fuller picture of how our modes perform

The first Linear Regression Model was created to help predict the price of a Sephora item in relation to a list of predictors, including rating, number_of_reviews, love, and different dummy variables indicating if an item is of a certain category or not. Continous variables were z-scored, which sets the variables on a similar scale. This model was explored in Question 2. In this question, we will be exploring how PCA will impact the same hypothesis if it is included in the model. Principal Component Analysis takes a dataset with a large range of predictors, and condenses it to somethings smaller and more managable, making it less computationaly expensive. We want to see how this model will behave with 80% of the original data.

To begin, we wanted to create a plot where we could find the optimal amount of Principal Components from the data that we want to include while keeping 90% original variance. This is done with a cumulative variance plot, where we plot the relationship between the variance and the PC's. When the y axis is 0.90, or 90% on the cumulative variance plot, the amount of PC's is 2. We can retain 2 of the components instead of using all 13.

Then, once the model construction is complete, I included some visuals to see how the model would perform. There are two plots, one that compares the predicted prices from Linear Regression's model with the true price, and one that compares Linear Regression's model with PCA's predictions to the true price. From the looks of it, both models are not the most accurate at predicting price. The Linear Regression model looks like it does a bit better, but not by much. It looks like this model has a tendency to predict prices around $50, even if it's not even close to being true. If these models were perfect at predicting the actual price, there would be a perfect diagonal line of data points in both visuals. The PCA model worsens the accuracy, as we can see in it's visual. It really struggles correctly classifying models with higher prices. This could possibly be because there are less expensive items sold at Sephora in comparison to other items.

After looking at the visuals, we wanted to explore some metrics to help discern the model's behavior, and if it is good at predicting price like we want it to. We looked at both the Linear Regression and Linear Regression with PCA's mean absolute error, mean squared error, and R-Squared metrics. Mean absolute error is a metric that measures the average distance between the true value and what the model predicts. It shows how off our model is. Mean squared error calculates the difference between what the model predicted in comparison to the true value, squared. Once again, it shows how off our model might be. Lastly, r-squared measures the accuracy of our model.

The MAE for the train and test set for the original Linear Regression model was not too off from each other. They were both at about 30. It is good because they are close to each other, indicating that the models are not overfit. However, 30 is a little high when we are measuring error. Same goes for the MAE of the PCA model. They are both at about 30 as well. It is good that the train and test set have the same number, but it is concerning that it is so high. On top of this, it is important to note that the MAE's between the original and the PCA model increases slightly. It would be ideal that the PCA metrics improve, but this is not the case.

Like the MAE units, the MSE units show that no overfitting is occuring. The units are similar between the train and test sets for both the original model and the PCA model. However, once again, they are both incredibly high. The PCA model does not improve this metric either. If anything, it increases it slightly.

Lastly, we have the R-Squared metrics for this model. This measures the accuracy of the model. For both the train and test sets for the two different models, we can see that all of the units are abysmally low. Typically, we aim for an R-Squared value of 1, indicating perfect accuracy. The PCA metrics show that PCA does not improve the accuracy of this model at all.

Based on these two models, I would say that performing PCA is unneccessary. It should help reduce excessive data and improve the model's performance, but this did not happen. This is reflected in the performance graph visuals, as well as the metrics. The Linear Regression metrics were poor to begin with, and the metrics do not improve when we look at their PCA counterparts. Seeing this lack of improvement, it would be wise to conclude that PCA would just waste lines of code for unsatisfactory results. We should re-evaluate our Linear Regression model and see what else we could do to improve results instead. It's reassuring that this model does not overfit, but it is completely inaccurate

1.8 Question 5 - Kylie

Using the Naive Bayes model, predict if an item has a marketing flag based on if the product is online only, limited edition, exclusive and is a limited time offer. How will the accuracy scores perform?

[57]:
```
sephoraNew = pd.get_dummies(sephora, columns = ["category"])
sephoraNew.head()
```

[57]:
```
   index       id          brand                          name  \
0      0  2218774  Acqua Di Parma  Blu Mediterraneo MINIATURE Set
1      1  2044816  Acqua Di Parma                        Colonia
2      2  1417567  Acqua Di Parma              Arancia di Capri
3      3  1417617  Acqua Di Parma              Mirto di Panarea
4      4  2218766  Acqua Di Parma         Colonia Miniature Set

            size  rating  number_of_reviews  love  price  value_price  …  \
0  5 x 0.16oz/5mL     4.0                  4  3002   66.0         75.0  …
1    0.7 oz/ 20 mL     4.5                 76  2700   66.0         66.0  …
2     5 oz/ 148 mL     4.5                 26  2600  180.0        180.0  …
3    2.5 oz/ 74 mL     4.5                 23  2900  120.0        120.0  …
4  5 x 0.16oz/5mL     3.5                  2   943   72.0         80.0  …

  category_Spa Tools  category_Sponges & Applicators category_Sunscreen  \
0                  0                               0                  0
```

```
1                  0                               0                  0
2                  0                               0                  0
3                  0                               0                  0
4                  0                               0                  0

  category_Teeth Whitening category_Tinted Moisturizer category_Toners  \
0                        0                           0               0
1                        0                           0               0
2                        0                           0               0
3                        0                           0               0
4                        0                           0               0

  category_Tweezers & Eyebrow Tools  category_Value & Gift Sets  \
0                                 0                           0
1                                 0                           0
2                                 0                           0
3                                 0                           0
4                                 0                           0

   category_Wellness  category_no category
0                  0                    0
1                  0                    0
2                  0                    0
3                  0                    0
4                  0                    0

[5 rows x 164 columns]
```

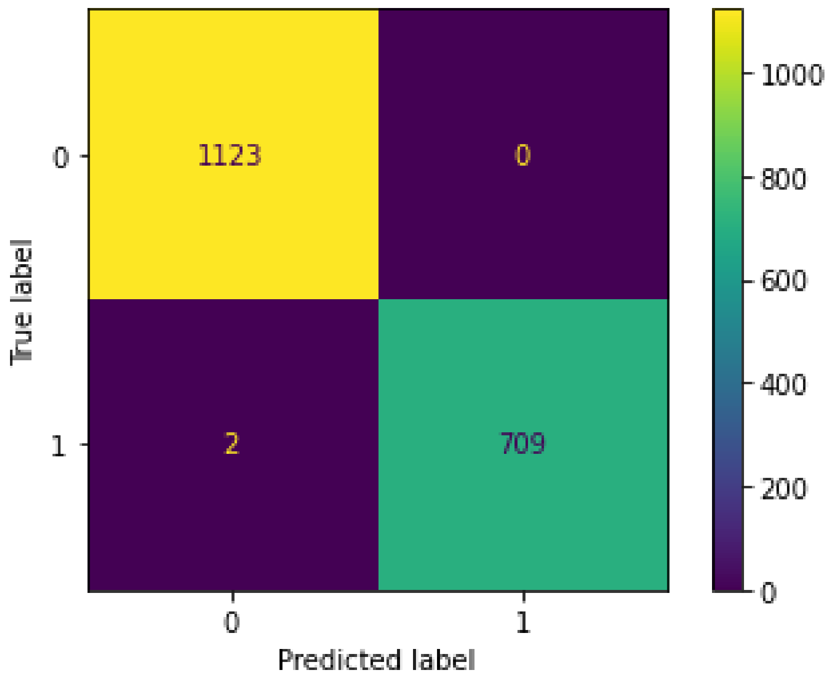
[58]:
```
sephora = pd.read_csv("https://raw.githubusercontent.com/KylieMullenex/cpsc392/
 ↪main/sephora_website_dataset.csv")
sephora = sephora.dropna()
sephora

sephoraNew = pd.get_dummies(sephora, columns = ["MarketingFlags"])
# sephoraNew.head()
```

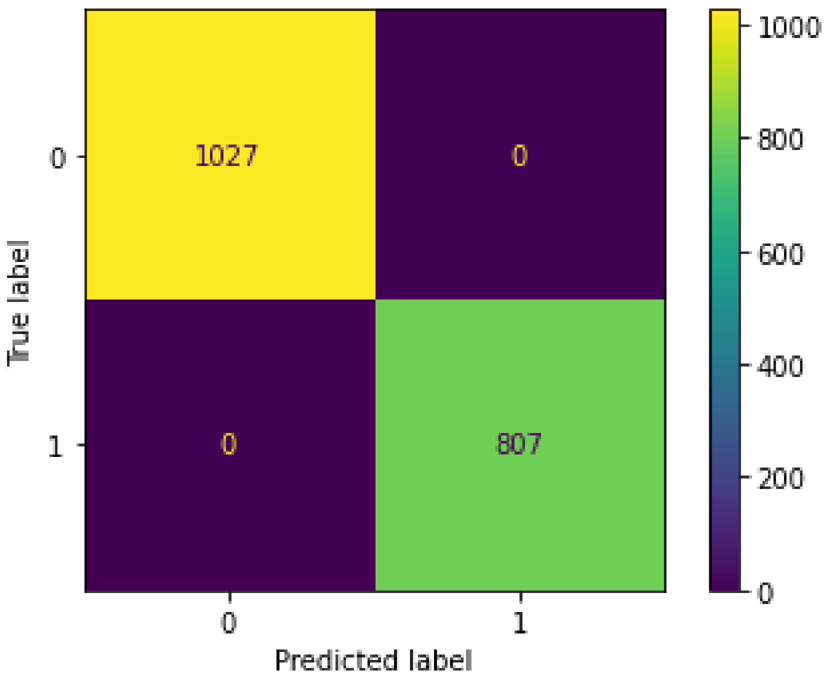
[59]:
```
predictors = ["online_only", "exclusive", "limited_edition",␣
 ↪"limited_time_offer"]
X = sephoraNew[predictors]
y = sephoraNew["MarketingFlags_True"]

kf = KFold(n_splits = 5)

acc_train = []
acc_test = []

nb = BernoulliNB()
```

```
for train,test in kf.split(X):
    X_train = X.iloc[train]
    X_test = X.iloc[test]
    y_train = y[train]
    y_test = y[test]

    nb.fit(X_train, y_train)

    acc_train.append(accuracy_score(y_train, nb.predict(X_train)))
    acc_test.append(accuracy_score(y_test, nb.predict(X_test)))

    plot_confusion_matrix(nb, X_test, y_test)

print("Test Acc: " + str(np.mean(acc_test)))
print("Train Acc: " + str(np.mean(acc_train)))


plot_roc_curve(nb, X_train, y_train)
plot_roc_curve(nb, X_test, y_test)
```

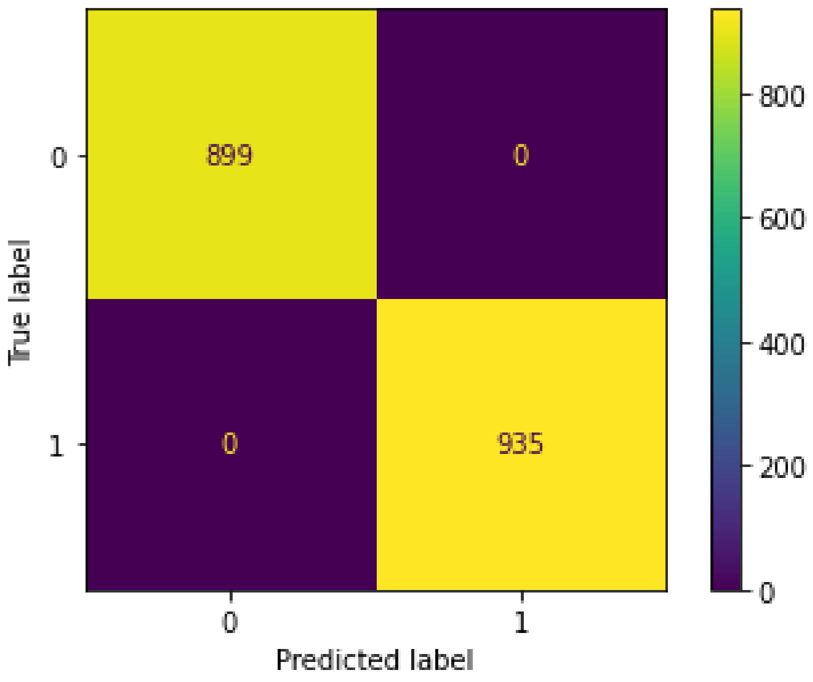
```
Test Acc: 0.9997818974918211
Train Acc: 0.9997818528860268
```

```
[59]: <sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x7fbd6f6c33a0>
```

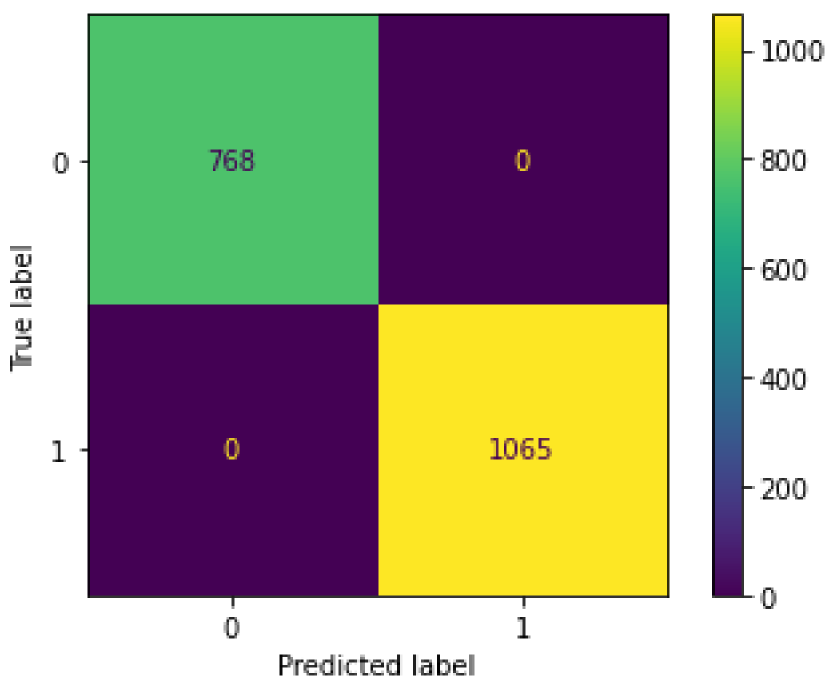


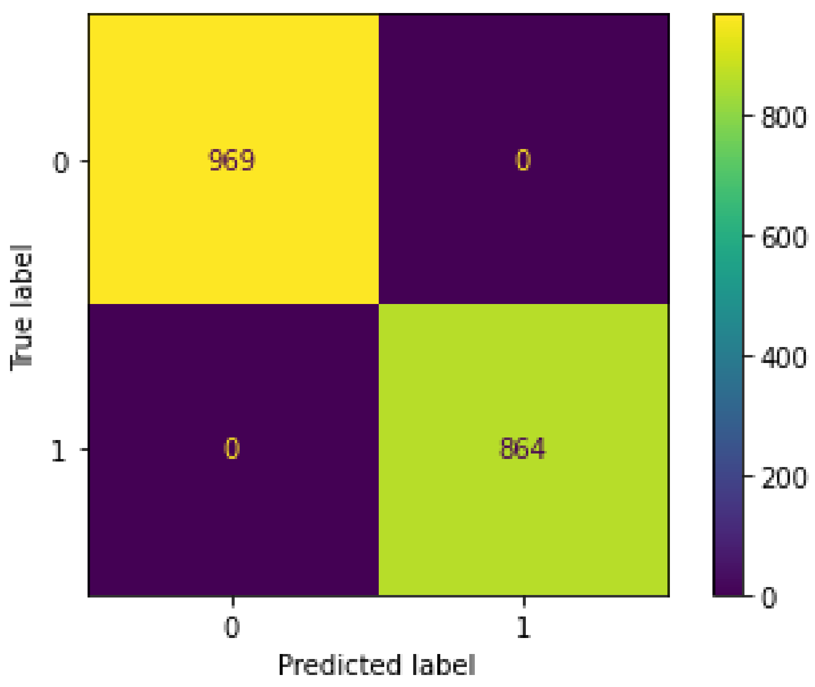


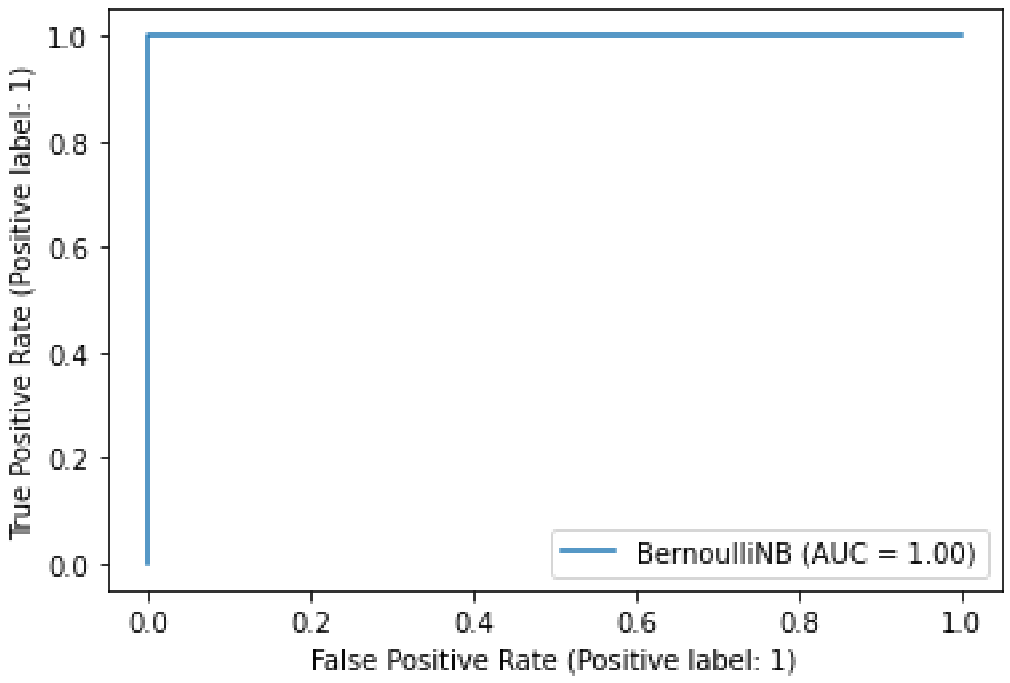
1027
0
0
807
True label
Predicted label
0
1
0
1
1000
800
600
400
200
0

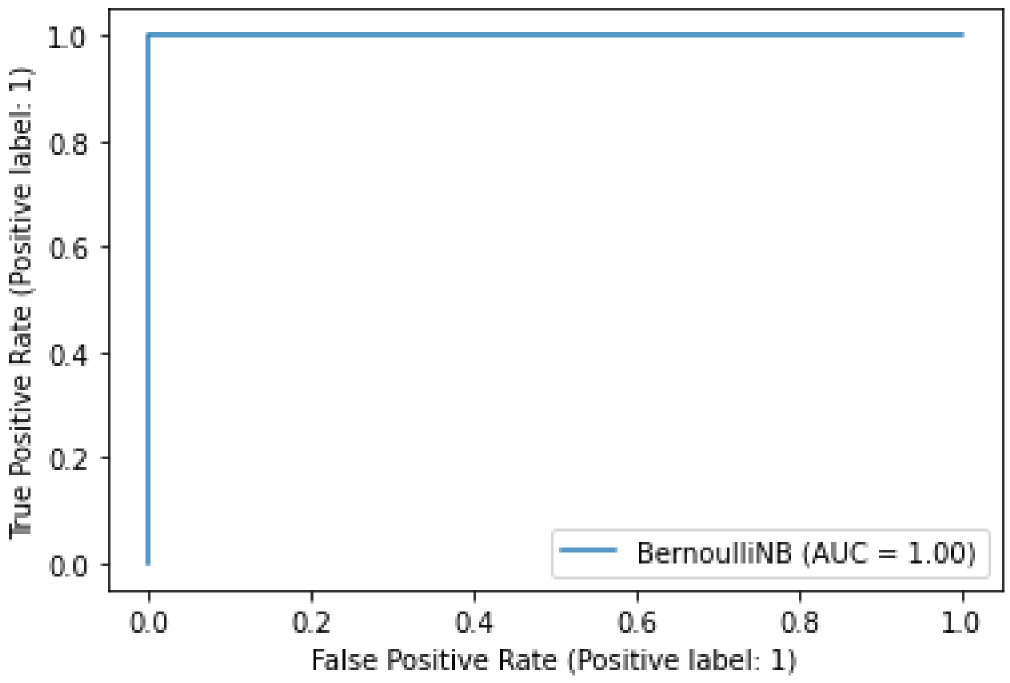


899
0
0
935
True label
Predicted label
0
1
0
1
800
600
400
200
0


768
0
0
1065
True label
Predicted label
0
1
0
1
1000
800
600
400
200
0


969
0
0
864
True label
Predicted label
0
1
0
1
800
600
400
200
0





Answer to Question 5:

Naive Bayes is a classifer algorithm that uses Bayes theorem to classify datapoints. Naive bayes is naive that it assumes that features of the data are independent of each other (we know this not to be true). Although this is normally an inaccurate assumption, because most of the time features are not independent of each other, it doesn't change the accuracy of the model too much so it is okay to be naive and use this classifier. For the bayes part of naive bayes, we calculate scores, or the probability of a datapoint being in a certain category. Whichever category/class has the highest score of that datapoint, we put the datapoint in that class. This naive bayes model does a very good job at predicting if an item has a marketing flag based on if the product is online only, limited edition, exclusive and is a limited time offer. The accuracy is the same score for both the train and test set, 99.97% accurate. This is an extremely accurate model and there is no overfitting occuring, or where the train accuracy is a lot higher than the test set accuracy.

A confusion matrix is a visual performance metrix that explains classification algorithims. It looks to see if there was a difference in the predictions of a datapoint to the actual value of the datapoint. In the confusion matrix there are 4 possible categoires of the matrix: true positive, true negative, false positive and false negative. True positive is where you predicted positive and it is positive. A false negative is where you predicted negative and it is negative. A false positive is when you predicted positive and it's false (it is negaitve). And a false negative is where you predicted negative and it's false (it is actually positive). Looking at the last confusion matrix you can see there are lots of true positives and true negatives meaning when it was true or false, the model was accurate in predicting that. It also has 0 false negatives or positivies, so it is not predicting any incorrect predictions. This confusion matrix shows that our model is doing very well for predicting marketing flags based on the selected predictors. Finally this question looked at an ROC graph to see how well the model is doing.

An ROC is curve is a graph showing the performance of a classification model. The curve looks at the true positive rate (how many times the model accuratley predicted true when it was true), and the false positive rate (how many times the model inaccuratley predicted it was true when it was false). The goal is to have an ROC closest to the top left corner or as close it can be to all true positives because the accuracy increases. In this model it is completley touching the top left corner and there are no false positives. This means our model is very accurate and good at predicting marketing flags from the selected predictors.

1.9 Question 6 - Kylie

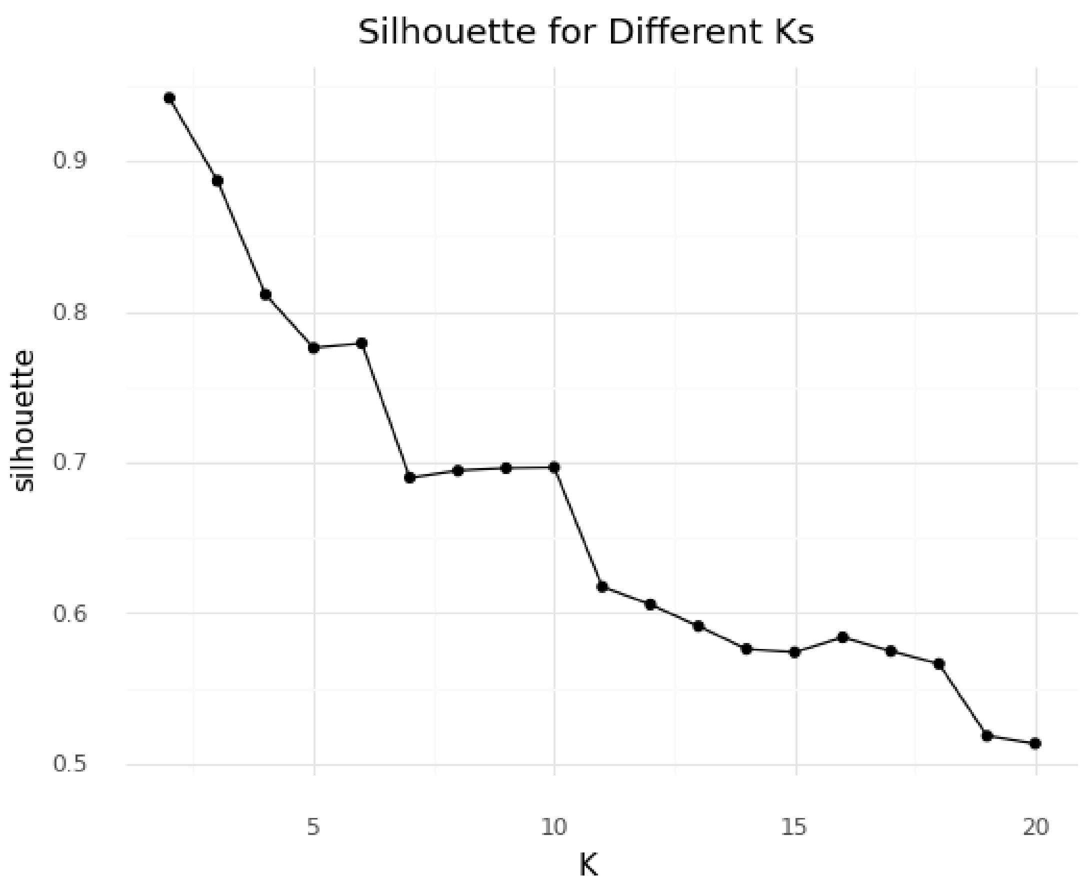
If we look at rating, number of reviews and value price, what clusters could emerge based on our model? What is the silhouette score? Describe what characterizes each cluster, and give an example of what type of cluster that product could exemplify.

[60]:
```
features = ["rating", "number_of_reviews", "value_price"]
X = sephoraNew[features]
# z = StandardScaler()

# X[features] = z.fit_transform(X[features])

# model
km = KMeans(n_clusters = 2)
km.fit(X[features])
```

```
membership = km.predict(X[features])

X["cluster"] = membership

print("The Silhouette score is: ", silhouette_score(X[features], membership))
```

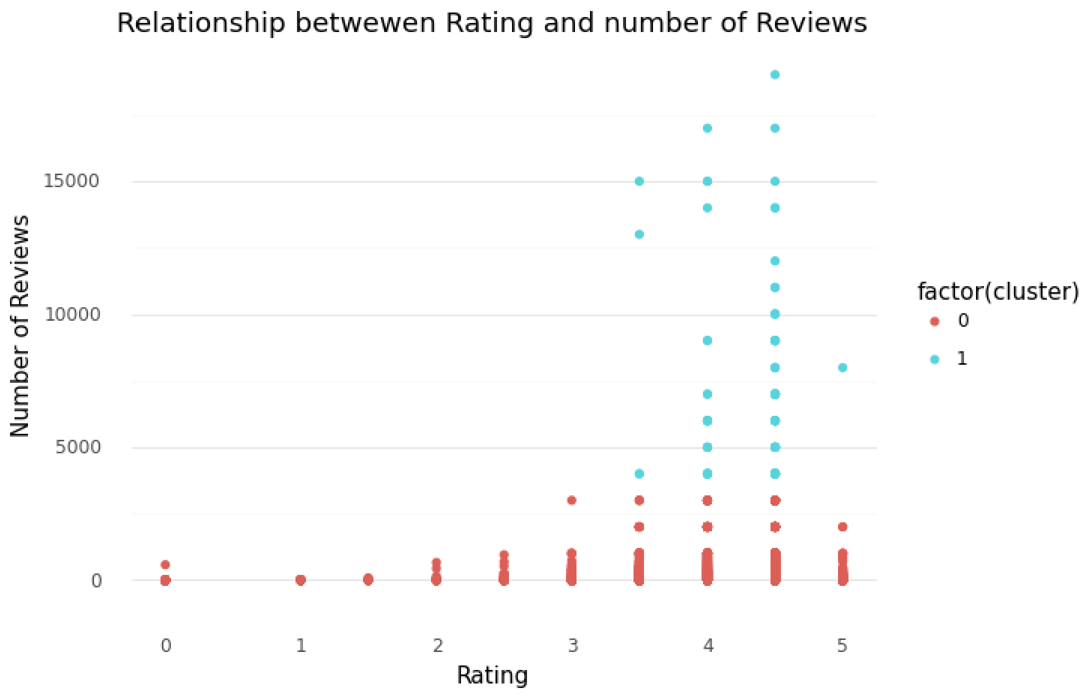
```
The Silhouette score is:  0.941853690519206
```

[61]:
```
ks = [2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]

sse =  []
sil = []

for k in ks:
  km = KMeans(n_clusters = k)
  km.fit(X[features])

  sse.append(km.inertia_)
  sil.append(silhouette_score(X[features], km.predict(X[features])))

sse_df = pd.DataFrame({"K": ks,
                       "sse": sse,
                       "silhouette": sil})


(ggplot(sse_df, aes(x = "K", y = "silhouette")) +
 geom_line() + geom_point() +
 theme_minimal() +
 labs(title = "Silhouette for Different Ks"))
```

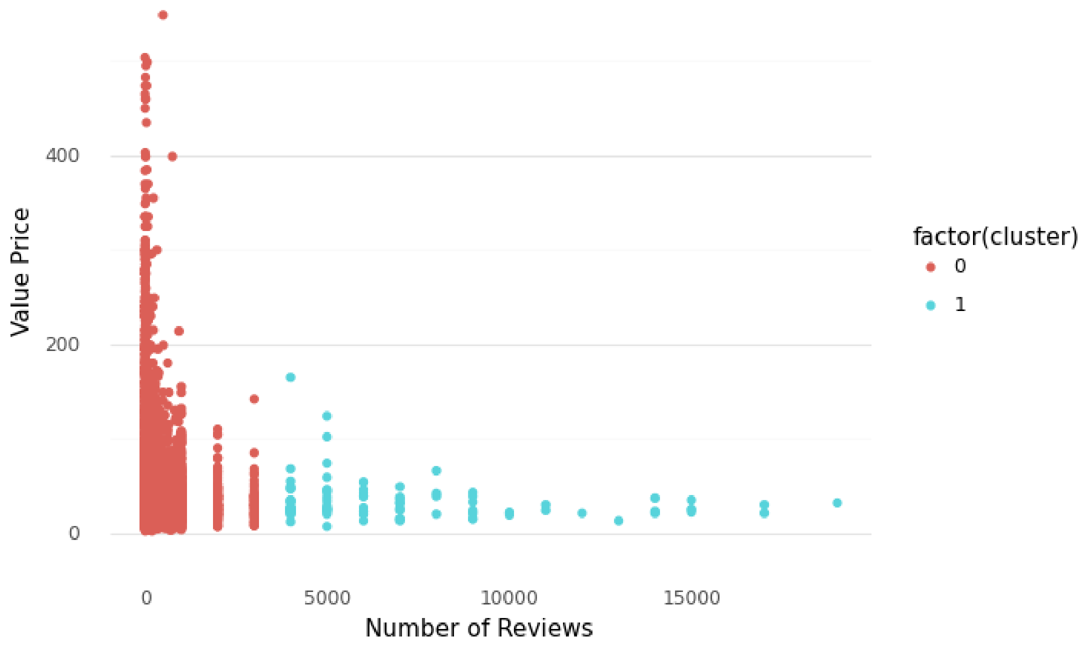


```
[61]: <ggplot: (8778223348845)>
```

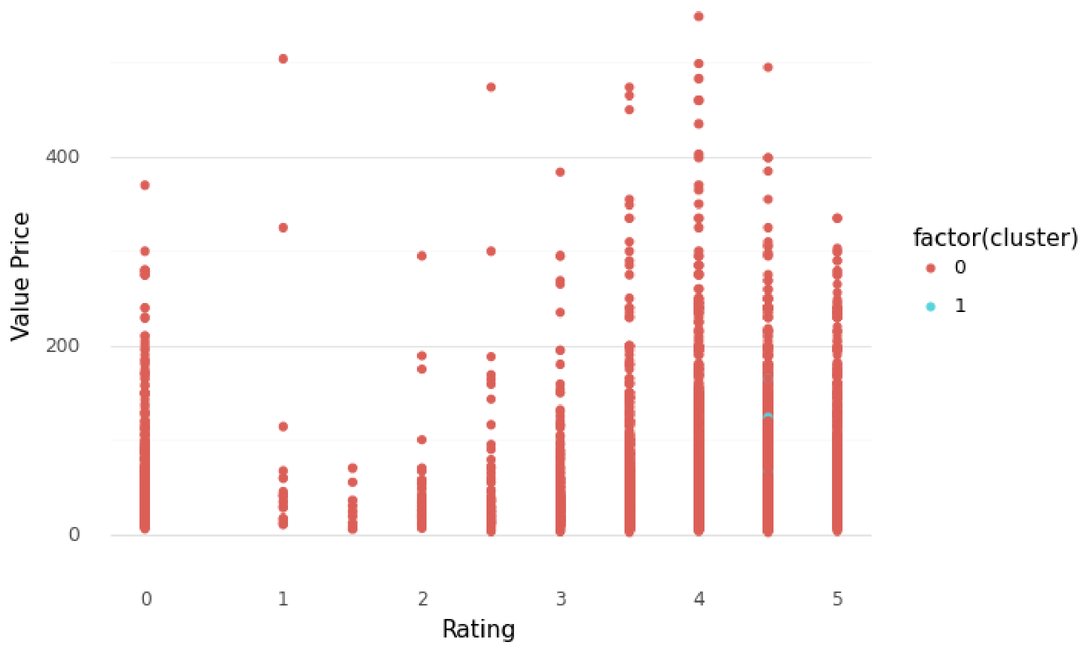
```
[62]: print((ggplot(X, aes(x = "rating", y = "number_of_reviews", color =␣
       ↪"factor(cluster)")) +
             geom_point() +
             theme_minimal() +
             theme(panel_grid_major_x = element_blank(),
                   panel_grid_minor_x = element_blank()) +
             labs(x = "Rating",
                  y = "Number of Reviews",
                  title = "Relationship betwewen Rating and number of Reviews")))

      print((ggplot(X, aes(x = "number_of_reviews", y = "value_price", color =␣
       ↪"factor(cluster)")) +
             geom_point() +
             theme_minimal() +
             theme(panel_grid_major_x = element_blank(),
                   panel_grid_minor_x = element_blank()) +
             labs(x = "Number of Reviews",
                  y = "Value Price",
                  title = "Relationship Between Number of Reviews and Value Price")))


      print((ggplot(X, aes(x = "rating", y =  "value_price", color =␣
       ↪"factor(cluster)")) +
             geom_point() +
             theme_minimal() +
             theme(panel_grid_major_x = element_blank(),
                   panel_grid_minor_x = element_blank()) +
             labs(x = "Rating",
                  y = "Value Price ",
                  title = "Relationship Between Rating and Value Price")))
```



Relationship Between Number of Reviews and Value Price



Relationship Between Rating and Value Price



Response to Question 6:

To look what clusters emerge for rating, number of review, and value price, Kmeans was used. Kmeans is an unsupervised machine learning that divides data into K clusters and assigns datapoints into each cluster that minimizes the total variance. Kmeans is hard assignment meaning a datapoint ca belong to one cluster only. A silhouette score is how we determine if the clustering techinque was good for the data. Silhouette scores measure cohesion and separation. Cohesion is defined as how similar datapoints in a cluster are to each other and separation is defined as how separate or distinct clusters are from each other. A good silhouette score is as close to 1 or -1 possible. Looking at different Ks 1 - 20 you can compare how good the silhouette score is. You can see that clustering the model into 2 categories/clusters results in the highest possible silhouette score of 0.9482. This is a good silhouette score and is very close to one which means the clustering algorithm for the data makes cohesive but separate clusters.

For this clustering I created three data visualizations to look at the clusters that emerge based on three predictors. The first graph looks at **rating and number of reviews**. The first cluster, or the cluster in red seems to be a wide range of ratings either from 0 to 5 but they all have low reviews. I would assume these are the products that not a lot of people use. For example someone bought a cologne didn't like it but didn't feel the need to review it, or someone bought a mascara loved it but didn't review it. These products are probably not bought that often and they can either be really good products but very few people review them or not great products and the reviews are minimal too. The second cluster that appears is for the products that have high rating either a 3 or 4 and high reviews. These would be the solid products that are consistently good, and lots of people rate it becaue they enjoy the product.

The second graph compares **number of reviews to value price**. Looking at the two clusters that emerge the first cluster seems to be products with a low to medium amount of reviews and medium to high value price. These would probably be products that not many people buy therefore they don't review. Probably due to the price not many people would buy them, therefore people wouldn't be able to reivew those products. The second cluster emerging would be those products that have a high number of reviews and a low value price. These are probably cheaper, affordable products that are "staples" for people to have and a lot of people like the products and buy them therefore they review it.

Finally the last graph compares **rating and value price.** Although the best amount of clusters to produce the highest silheoutte score was 2, there is only 1 cluster that emerged for rating and value price. That is probably because there is no way to cluster these groups because it varies so much.

[63]:
```
# Download as PDF with Jupyter Notebook

# doesn't show this cells output when downloading PDF
!pip install gwpy &> /dev/null

# installing necessary files
!apt-get install texlive texlive-xetex texlive-latex-extra pandoc
!sudo apt-get update
```

```
!sudo apt-get install texlive-xetex texlive-fonts-recommended␣
↪texlive-plain-generic

# installing pypandoc
!pip install pypandoc

# connecting your google drive
from google.colab import drive
drive.mount('/content/drive')

# copying your file over. Change "Class6-Completed.ipynb" to whatever your file␣
↪is called (see top of notebook)
!cp "drive/My Drive/Colab Notebooks/FinalProject1.ipynb" ./

# Again, replace "Class6-Completed.ipynb" to whatever your file is called (see␣
↪top of notebook)
!jupyter nbconvert --to PDF "FinalProject1.ipynb"
```

```
Reading package lists… Done
Building dependency tree
Reading state information… Done
pandoc is already the newest version (1.19.2.4~dfsg-1build4).
pandoc set to manually installed.
The following package was automatically installed and is no longer required:
  libnvidia-common-460
Use 'apt autoremove' to remove it.
The following additional packages will be installed:
  fonts-droid-fallback fonts-lato fonts-lmodern fonts-noto-mono fonts-texgyre
  javascript-common libcupsfilters1 libcupsimage2 libgs9 libgs9-common
  libijs-0.35 libjbig2dec0 libjs-jquery libkpathsea6 libpotrace0 libptexenc1
  libruby2.5 libsynctex1 libtexlua52 libtexluajit2 libzzip-0-13 lmodern
  poppler-data preview-latex-style rake ruby ruby-did-you-mean ruby-minitest
  ruby-net-telnet ruby-power-assert ruby-test-unit ruby2.5
  rubygems-integration t1utils tex-common tex-gyre texlive-base
  texlive-binaries texlive-fonts-recommended texlive-latex-base
  texlive-latex-recommended texlive-pictures texlive-plain-generic tipa
Suggested packages:
  fonts-noto apache2 | lighttpd | httpd poppler-utils ghostscript
  fonts-japanese-mincho | fonts-ipafont-mincho fonts-japanese-gothic
  | fonts-ipafont-gothic fonts-arphic-ukai fonts-arphic-uming fonts-nanum ri
  ruby-dev bundler debhelper gv | postscript-viewer perl-tk xpdf-reader
  | pdf-viewer texlive-fonts-recommended-doc texlive-latex-base-doc
  python-pygments icc-profiles libfile-which-perl
  libspreadsheet-parseexcel-perl texlive-latex-extra-doc
  texlive-latex-recommended-doc texlive-pstricks dot2tex prerex ruby-tcltk
  | libtcltk-ruby texlive-pictures-doc vprerex
The following NEW packages will be installed:
  fonts-droid-fallback fonts-lato fonts-lmodern fonts-noto-mono fonts-texgyre
  javascript-common libcupsfilters1 libcupsimage2 libgs9 libgs9-common
  libijs-0.35 libjbig2dec0 libjs-jquery libkpathsea6 libpotrace0 libptexenc1
  libruby2.5 libsynctex1 libtexlua52 libtexluajit2 libzzip-0-13 lmodern
  poppler-data preview-latex-style rake ruby ruby-did-you-mean ruby-minitest
  ruby-net-telnet ruby-power-assert ruby-test-unit ruby2.5
  rubygems-integration t1utils tex-common tex-gyre texlive texlive-base
  texlive-binaries texlive-fonts-recommended texlive-latex-base
  texlive-latex-extra texlive-latex-recommended texlive-pictures
  texlive-plain-generic texlive-xetex tipa
0 upgraded, 47 newly installed, 0 to remove and 20 not upgraded.
Need to get 146 MB of archives.
After this operation, 460 MB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu bionic/main amd64 fonts-droid-fallback
all 1:6.0.1r16-1.1 [1,805 kB]
Get:2 http://archive.ubuntu.com/ubuntu bionic/main amd64 fonts-lato all 2.0-2
[2,698 kB]
Get:3 http://archive.ubuntu.com/ubuntu bionic/main amd64 poppler-data all
0.4.8-2 [1,479 kB]
Get:4 http://archive.ubuntu.com/ubuntu bionic/main amd64 tex-common all 6.09
[33.0 kB]
Get:5 http://archive.ubuntu.com/ubuntu bionic/main amd64 fonts-lmodern all
2.004.5-3 [4,551 kB]
Get:6 http://archive.ubuntu.com/ubuntu bionic/main amd64 fonts-noto-mono all
20171026-2 [75.5 kB]
Get:7 http://archive.ubuntu.com/ubuntu bionic/universe amd64 fonts-texgyre all
20160520-1 [8,761 kB]
Get:8 http://archive.ubuntu.com/ubuntu bionic/main amd64 javascript-common all
11 [6,066 B]
Get:9 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libcupsfilters1
amd64 1.20.2-0ubuntu3.1 [108 kB]
Get:10 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libcupsimage2
amd64 2.2.7-1ubuntu2.9 [18.6 kB]
Get:11 http://archive.ubuntu.com/ubuntu bionic/main amd64 libijs-0.35 amd64
0.35-13 [15.5 kB]
Get:12 http://archive.ubuntu.com/ubuntu bionic/main amd64 libjbig2dec0 amd64
0.13-6 [55.9 kB]
Get:13 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libgs9-common
all 9.26~dfsg+0-0ubuntu0.18.04.17 [5,092 kB]
Get:14 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libgs9 amd64
9.26~dfsg+0-0ubuntu0.18.04.17 [2,267 kB]
Get:15 http://archive.ubuntu.com/ubuntu bionic/main amd64 libjs-jquery all
3.2.1-1 [152 kB]
Get:16 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libkpathsea6
amd64 2017.20170613.44572-8ubuntu0.1 [54.9 kB]
Get:17 http://archive.ubuntu.com/ubuntu bionic/main amd64 libpotrace0 amd64
1.14-2 [17.4 kB]
Get:18 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libptexenc1
```

```
amd64 2017.20170613.44572-8ubuntu0.1 [34.5 kB]
Get:19 http://archive.ubuntu.com/ubuntu bionic/main amd64 rubygems-integration
all 1.11 [4,994 B]
Get:20 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 ruby2.5 amd64
2.5.1-1ubuntu1.12 [48.6 kB]
Get:21 http://archive.ubuntu.com/ubuntu bionic/main amd64 ruby amd64 1:2.5.1
[5,712 B]
Get:22 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 rake all
12.3.1-1ubuntu0.1 [44.9 kB]
Get:23 http://archive.ubuntu.com/ubuntu bionic/main amd64 ruby-did-you-mean all
1.2.0-2 [9,700 B]
Get:24 http://archive.ubuntu.com/ubuntu bionic/main amd64 ruby-minitest all
5.10.3-1 [38.6 kB]
Get:25 http://archive.ubuntu.com/ubuntu bionic/main amd64 ruby-net-telnet all
0.1.1-2 [12.6 kB]
Get:26 http://archive.ubuntu.com/ubuntu bionic/main amd64 ruby-power-assert all
0.3.0-1 [7,952 B]
Get:27 http://archive.ubuntu.com/ubuntu bionic/main amd64 ruby-test-unit all
3.2.5-1 [61.1 kB]
Get:28 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libruby2.5
amd64 2.5.1-1ubuntu1.12 [3,073 kB]
Get:29 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libsynctex1
amd64 2017.20170613.44572-8ubuntu0.1 [41.4 kB]
Get:30 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libtexlua52
amd64 2017.20170613.44572-8ubuntu0.1 [91.2 kB]
Get:31 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libtexluajit2
amd64 2017.20170613.44572-8ubuntu0.1 [230 kB]
Get:32 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libzzip-0-13
amd64 0.13.62-3.1ubuntu0.18.04.1 [26.0 kB]
Get:33 http://archive.ubuntu.com/ubuntu bionic/main amd64 lmodern all 2.004.5-3
[9,631 kB]
Get:34 http://archive.ubuntu.com/ubuntu bionic/main amd64 preview-latex-style
all 11.91-1ubuntu1 [185 kB]
Get:35 http://archive.ubuntu.com/ubuntu bionic/main amd64 t1utils amd64 1.41-2
[56.0 kB]
Get:36 http://archive.ubuntu.com/ubuntu bionic/universe amd64 tex-gyre all
20160520-1 [4,998 kB]
Get:37 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 texlive-
binaries amd64 2017.20170613.44572-8ubuntu0.1 [8,179 kB]
Get:38 http://archive.ubuntu.com/ubuntu bionic/main amd64 texlive-base all
2017.20180305-1 [18.7 MB]
Get:39 http://archive.ubuntu.com/ubuntu bionic/universe amd64 texlive-fonts-
recommended all 2017.20180305-1 [5,262 kB]
Get:40 http://archive.ubuntu.com/ubuntu bionic/main amd64 texlive-latex-base all
2017.20180305-1 [951 kB]
Get:41 http://archive.ubuntu.com/ubuntu bionic/main amd64 texlive-latex-
recommended all 2017.20180305-1 [14.9 MB]
Get:42 http://archive.ubuntu.com/ubuntu bionic/universe amd64 texlive all
```

```
2017.20180305-1 [14.4 kB]
Get:43 http://archive.ubuntu.com/ubuntu bionic/universe amd64 texlive-pictures
all 2017.20180305-1 [4,026 kB]
Get:44 http://archive.ubuntu.com/ubuntu bionic/universe amd64 texlive-latex-
extra all 2017.20180305-2 [10.6 MB]
Get:45 http://archive.ubuntu.com/ubuntu bionic/universe amd64 texlive-plain-
generic all 2017.20180305-2 [23.6 MB]
Get:46 http://archive.ubuntu.com/ubuntu bionic/universe amd64 tipa all 2:1.3-20
[2,978 kB]
Get:47 http://archive.ubuntu.com/ubuntu bionic/universe amd64 texlive-xetex all
2017.20180305-1 [10.7 MB]
Fetched 146 MB in 3s (55.8 MB/s)
Extracting templates from packages: 100%
Preconfiguring packages …
Selecting previously unselected package fonts-droid-fallback.
(Reading database … 124013 files and directories currently installed.)
Preparing to unpack …/00-fonts-droid-fallback_1%3a6.0.1r16-1.1_all.deb …
Unpacking fonts-droid-fallback (1:6.0.1r16-1.1) …
Selecting previously unselected package fonts-lato.
Preparing to unpack …/01-fonts-lato_2.0-2_all.deb …
Unpacking fonts-lato (2.0-2) …
Selecting previously unselected package poppler-data.
Preparing to unpack …/02-poppler-data_0.4.8-2_all.deb …
Unpacking poppler-data (0.4.8-2) …
Selecting previously unselected package tex-common.
Preparing to unpack …/03-tex-common_6.09_all.deb …
Unpacking tex-common (6.09) …
Selecting previously unselected package fonts-lmodern.
Preparing to unpack …/04-fonts-lmodern_2.004.5-3_all.deb …
Unpacking fonts-lmodern (2.004.5-3) …
Selecting previously unselected package fonts-noto-mono.
Preparing to unpack …/05-fonts-noto-mono_20171026-2_all.deb …
Unpacking fonts-noto-mono (20171026-2) …
Selecting previously unselected package fonts-texgyre.
Preparing to unpack …/06-fonts-texgyre_20160520-1_all.deb …
Unpacking fonts-texgyre (20160520-1) …
Selecting previously unselected package javascript-common.
Preparing to unpack …/07-javascript-common_11_all.deb …
Unpacking javascript-common (11) …
Selecting previously unselected package libcupsfilters1:amd64.
Preparing to unpack …/08-libcupsfilters1_1.20.2-0ubuntu3.1_amd64.deb …
Unpacking libcupsfilters1:amd64 (1.20.2-0ubuntu3.1) …
Selecting previously unselected package libcupsimage2:amd64.
Preparing to unpack …/09-libcupsimage2_2.2.7-1ubuntu2.9_amd64.deb …
Unpacking libcupsimage2:amd64 (2.2.7-1ubuntu2.9) …
Selecting previously unselected package libijs-0.35:amd64.
Preparing to unpack …/10-libijs-0.35_0.35-13_amd64.deb …
Unpacking libijs-0.35:amd64 (0.35-13) …
```

```
Selecting previously unselected package libjbig2dec0:amd64.
Preparing to unpack …/11-libjbig2dec0_0.13-6_amd64.deb …
Unpacking libjbig2dec0:amd64 (0.13-6) …
Selecting previously unselected package libgs9-common.
Preparing to unpack …/12-libgs9-common_9.26~dfsg+0-0ubuntu0.18.04.17_all.deb
…
Unpacking libgs9-common (9.26~dfsg+0-0ubuntu0.18.04.17) …
Selecting previously unselected package libgs9:amd64.
Preparing to unpack …/13-libgs9_9.26~dfsg+0-0ubuntu0.18.04.17_amd64.deb …
Unpacking libgs9:amd64 (9.26~dfsg+0-0ubuntu0.18.04.17) …
Selecting previously unselected package libjs-jquery.
Preparing to unpack …/14-libjs-jquery_3.2.1-1_all.deb …
Unpacking libjs-jquery (3.2.1-1) …
Selecting previously unselected package libkpathsea6:amd64.
Preparing to unpack …/15-libkpathsea6_2017.20170613.44572-8ubuntu0.1_amd64.deb
…
Unpacking libkpathsea6:amd64 (2017.20170613.44572-8ubuntu0.1) …
Selecting previously unselected package libpotrace0.
Preparing to unpack …/16-libpotrace0_1.14-2_amd64.deb …
Unpacking libpotrace0 (1.14-2) …
Selecting previously unselected package libptexenc1:amd64.
Preparing to unpack …/17-libptexenc1_2017.20170613.44572-8ubuntu0.1_amd64.deb
…
Unpacking libptexenc1:amd64 (2017.20170613.44572-8ubuntu0.1) …
Selecting previously unselected package rubygems-integration.
Preparing to unpack …/18-rubygems-integration_1.11_all.deb …
Unpacking rubygems-integration (1.11) …
Selecting previously unselected package ruby2.5.
Preparing to unpack …/19-ruby2.5_2.5.1-1ubuntu1.12_amd64.deb …
Unpacking ruby2.5 (2.5.1-1ubuntu1.12) …
Selecting previously unselected package ruby.
Preparing to unpack …/20-ruby_1%3a2.5.1_amd64.deb …
Unpacking ruby (1:2.5.1) …
Selecting previously unselected package rake.
Preparing to unpack …/21-rake_12.3.1-1ubuntu0.1_all.deb …
Unpacking rake (12.3.1-1ubuntu0.1) …
Selecting previously unselected package ruby-did-you-mean.
Preparing to unpack …/22-ruby-did-you-mean_1.2.0-2_all.deb …
Unpacking ruby-did-you-mean (1.2.0-2) …
Selecting previously unselected package ruby-minitest.
Preparing to unpack …/23-ruby-minitest_5.10.3-1_all.deb …
Unpacking ruby-minitest (5.10.3-1) …
Selecting previously unselected package ruby-net-telnet.
Preparing to unpack …/24-ruby-net-telnet_0.1.1-2_all.deb …
Unpacking ruby-net-telnet (0.1.1-2) …
Selecting previously unselected package ruby-power-assert.
Preparing to unpack …/25-ruby-power-assert_0.3.0-1_all.deb …
Unpacking ruby-power-assert (0.3.0-1) …
```

```
Selecting previously unselected package ruby-test-unit.
Preparing to unpack …/26-ruby-test-unit_3.2.5-1_all.deb …
Unpacking ruby-test-unit (3.2.5-1) …
Selecting previously unselected package libruby2.5:amd64.
Preparing to unpack …/27-libruby2.5_2.5.1-1ubuntu1.12_amd64.deb …
Unpacking libruby2.5:amd64 (2.5.1-1ubuntu1.12) …
Selecting previously unselected package libsynctex1:amd64.
Preparing to unpack …/28-libsynctex1_2017.20170613.44572-8ubuntu0.1_amd64.deb
…
Unpacking libsynctex1:amd64 (2017.20170613.44572-8ubuntu0.1) …
Selecting previously unselected package libtexlua52:amd64.
Preparing to unpack …/29-libtexlua52_2017.20170613.44572-8ubuntu0.1_amd64.deb
…
Unpacking libtexlua52:amd64 (2017.20170613.44572-8ubuntu0.1) …
Selecting previously unselected package libtexluajit2:amd64.
Preparing to unpack
…/30-libtexluajit2_2017.20170613.44572-8ubuntu0.1_amd64.deb …
Unpacking libtexluajit2:amd64 (2017.20170613.44572-8ubuntu0.1) …
Selecting previously unselected package libzzip-0-13:amd64.
Preparing to unpack …/31-libzzip-0-13_0.13.62-3.1ubuntu0.18.04.1_amd64.deb …
Unpacking libzzip-0-13:amd64 (0.13.62-3.1ubuntu0.18.04.1) …
Selecting previously unselected package lmodern.
Preparing to unpack …/32-lmodern_2.004.5-3_all.deb …
Unpacking lmodern (2.004.5-3) …
Selecting previously unselected package preview-latex-style.
Preparing to unpack …/33-preview-latex-style_11.91-1ubuntu1_all.deb …
Unpacking preview-latex-style (11.91-1ubuntu1) …
Selecting previously unselected package t1utils.
Preparing to unpack …/34-t1utils_1.41-2_amd64.deb …
Unpacking t1utils (1.41-2) …
Selecting previously unselected package tex-gyre.
Preparing to unpack …/35-tex-gyre_20160520-1_all.deb …
Unpacking tex-gyre (20160520-1) …
Selecting previously unselected package texlive-binaries.
Preparing to unpack …/36-texlive-
binaries_2017.20170613.44572-8ubuntu0.1_amd64.deb …
Unpacking texlive-binaries (2017.20170613.44572-8ubuntu0.1) …
Selecting previously unselected package texlive-base.
Preparing to unpack …/37-texlive-base_2017.20180305-1_all.deb …
Unpacking texlive-base (2017.20180305-1) …
Selecting previously unselected package texlive-fonts-recommended.
Preparing to unpack …/38-texlive-fonts-recommended_2017.20180305-1_all.deb …
Unpacking texlive-fonts-recommended (2017.20180305-1) …
Selecting previously unselected package texlive-latex-base.
Preparing to unpack …/39-texlive-latex-base_2017.20180305-1_all.deb …
Unpacking texlive-latex-base (2017.20180305-1) …
Selecting previously unselected package texlive-latex-recommended.
Preparing to unpack …/40-texlive-latex-recommended_2017.20180305-1_all.deb …
```

```
Unpacking texlive-latex-recommended (2017.20180305-1) …
Selecting previously unselected package texlive.
Preparing to unpack …/41-texlive_2017.20180305-1_all.deb …
Unpacking texlive (2017.20180305-1) …
Selecting previously unselected package texlive-pictures.
Preparing to unpack …/42-texlive-pictures_2017.20180305-1_all.deb …
Unpacking texlive-pictures (2017.20180305-1) …
Selecting previously unselected package texlive-latex-extra.
Preparing to unpack …/43-texlive-latex-extra_2017.20180305-2_all.deb …
Unpacking texlive-latex-extra (2017.20180305-2) …
Selecting previously unselected package texlive-plain-generic.
Preparing to unpack …/44-texlive-plain-generic_2017.20180305-2_all.deb …
Unpacking texlive-plain-generic (2017.20180305-2) …
Selecting previously unselected package tipa.
Preparing to unpack …/45-tipa_2%3a1.3-20_all.deb …
Unpacking tipa (2:1.3-20) …
Selecting previously unselected package texlive-xetex.
Preparing to unpack …/46-texlive-xetex_2017.20180305-1_all.deb …
Unpacking texlive-xetex (2017.20180305-1) …
Setting up libgs9-common (9.26~dfsg+0-0ubuntu0.18.04.17) …
Setting up libkpathsea6:amd64 (2017.20170613.44572-8ubuntu0.1) …
Setting up libjs-jquery (3.2.1-1) …
Setting up libtexlua52:amd64 (2017.20170613.44572-8ubuntu0.1) …
Setting up fonts-droid-fallback (1:6.0.1r16-1.1) …
Setting up libsynctex1:amd64 (2017.20170613.44572-8ubuntu0.1) …
Setting up libptexenc1:amd64 (2017.20170613.44572-8ubuntu0.1) …
Setting up tex-common (6.09) …
update-language: texlive-base not installed and configured, doing nothing!
Setting up poppler-data (0.4.8-2) …
Setting up tex-gyre (20160520-1) …
Setting up preview-latex-style (11.91-1ubuntu1) …
Setting up fonts-texgyre (20160520-1) …
Setting up fonts-noto-mono (20171026-2) …
Setting up fonts-lato (2.0-2) …
Setting up libcupsfilters1:amd64 (1.20.2-0ubuntu3.1) …
Setting up libcupsimage2:amd64 (2.2.7-1ubuntu2.9) …
Setting up libjbig2dec0:amd64 (0.13-6) …
Setting up ruby-did-you-mean (1.2.0-2) …
Setting up t1utils (1.41-2) …
Setting up ruby-net-telnet (0.1.1-2) …
Setting up libijs-0.35:amd64 (0.35-13) …
Setting up rubygems-integration (1.11) …
Setting up libpotrace0 (1.14-2) …
Setting up javascript-common (11) …
Setting up ruby-minitest (5.10.3-1) …
Setting up libzzip-0-13:amd64 (0.13.62-3.1ubuntu0.18.04.1) …
Setting up libgs9:amd64 (9.26~dfsg+0-0ubuntu0.18.04.17) …
Setting up libtexluajit2:amd64 (2017.20170613.44572-8ubuntu0.1) …
Setting up fonts-lmodern (2.004.5-3) …
Setting up ruby-power-assert (0.3.0-1) …
Setting up texlive-binaries (2017.20170613.44572-8ubuntu0.1) …
update-alternatives: using /usr/bin/xdvi-xaw to provide /usr/bin/xdvi.bin
(xdvi.bin) in auto mode
update-alternatives: using /usr/bin/bibtex.original to provide /usr/bin/bibtex
(bibtex) in auto mode
Setting up texlive-base (2017.20180305-1) …
mktexlsr: Updating /var/lib/texmf/ls-R-TEXLIVEDIST…
mktexlsr: Updating /var/lib/texmf/ls-R-TEXMFMAIN…
mktexlsr: Updating /var/lib/texmf/ls-R…
mktexlsr: Done.
tl-paper: setting paper size for dvips to a4:
/var/lib/texmf/dvips/config/config-paper.ps
tl-paper: setting paper size for dvipdfmx to a4:
/var/lib/texmf/dvipdfmx/dvipdfmx-paper.cfg
tl-paper: setting paper size for xdvi to a4: /var/lib/texmf/xdvi/XDvi-paper
tl-paper: setting paper size for pdftex to a4:
/var/lib/texmf/tex/generic/config/pdftexconfig.tex
Setting up texlive-fonts-recommended (2017.20180305-1) …
Setting up texlive-plain-generic (2017.20180305-2) …
Setting up texlive-latex-base (2017.20180305-1) …
Setting up lmodern (2.004.5-3) …
Setting up texlive-latex-recommended (2017.20180305-1) …
Setting up texlive-pictures (2017.20180305-1) …
Setting up tipa (2:1.3-20) …
Regenerating '/var/lib/texmf/fmtutil.cnf-DEBIAN'… done.
Regenerating '/var/lib/texmf/fmtutil.cnf-TEXLIVEDIST'… done.
update-fmtutil has updated the following file(s):
        /var/lib/texmf/fmtutil.cnf-DEBIAN
        /var/lib/texmf/fmtutil.cnf-TEXLIVEDIST
If you want to activate the changes in the above file(s),
you should run fmtutil-sys or fmtutil.
Setting up texlive (2017.20180305-1) …
Setting up texlive-latex-extra (2017.20180305-2) …
Setting up texlive-xetex (2017.20180305-1) …
Setting up ruby2.5 (2.5.1-1ubuntu1.12) …
Setting up ruby (1:2.5.1) …
Setting up ruby-test-unit (3.2.5-1) …
Setting up rake (12.3.1-1ubuntu0.1) …
Setting up libruby2.5:amd64 (2.5.1-1ubuntu1.12) …
Processing triggers for mime-support (3.60ubuntu1) …
Processing triggers for libc-bin (2.27-3ubuntu1.6) …
Processing triggers for man-db (2.8.3-2ubuntu0.1) …
Processing triggers for fontconfig (2.12.6-0ubuntu2) …
Processing triggers for tex-common (6.09) …
Running updmap-sys. This may take some time… done.
Running mktexlsr /var/lib/texmf … done.
```

```
Building format(s) --all.
        This may take some time… done.
Get:1 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Hit:2 http://archive.ubuntu.com/ubuntu bionic InRelease
Get:3 http://archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:4 https://cloud.r-project.org/bin/linux/ubuntu bionic-cran40/ InRelease
[3,626 B]
Ign:5 https://developer.download.nvidia.com/compute/machine-
learning/repos/ubuntu1804/x86_64  InRelease
Get:6 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64
InRelease [1,581 B]
Hit:7 http://ppa.launchpad.net/c2d4u.team/c2d4u4.0+/ubuntu bionic InRelease
Get:8 http://archive.ubuntu.com/ubuntu bionic-backports InRelease [83.3 kB]
Hit:9 https://developer.download.nvidia.com/compute/machine-
learning/repos/ubuntu1804/x86_64  Release
Hit:10 http://ppa.launchpad.net/cran/libgit2/ubuntu bionic InRelease
Get:11 http://security.ubuntu.com/ubuntu bionic-security/restricted amd64
Packages [1,311 kB]
Hit:12 http://ppa.launchpad.net/deadsnakes/ppa/ubuntu bionic InRelease
Get:13 http://archive.ubuntu.com/ubuntu bionic-updates/restricted amd64 Packages
[1,352 kB]
Get:14 http://security.ubuntu.com/ubuntu bionic-security/universe amd64 Packages
[1,567 kB]
Get:15 http://security.ubuntu.com/ubuntu bionic-security/main amd64 Packages
[3,099 kB]
Hit:16 http://ppa.launchpad.net/graphics-drivers/ppa/ubuntu bionic InRelease
Get:17 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 Packages
[3,524 kB]
Get:18
https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64
Packages [1,073 kB]
Get:19 http://archive.ubuntu.com/ubuntu bionic-updates/universe amd64 Packages
[2,342 kB]
Fetched 14.5 MB in 2s (6,129 kB/s)
Reading package lists… Done
Reading package lists… Done
Building dependency tree
Reading state information… Done
texlive-fonts-recommended is already the newest version (2017.20180305-1).
texlive-fonts-recommended set to manually installed.
texlive-plain-generic is already the newest version (2017.20180305-2).
texlive-plain-generic set to manually installed.
texlive-xetex is already the newest version (2017.20180305-1).
The following package was automatically installed and is no longer required:
  libnvidia-common-460
Use 'sudo apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 34 not upgraded.
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
```

```
wheels/public/simple/
Collecting pypandoc
  Downloading pypandoc-1.10-py3-none-any.whl (20 kB)
Installing collected packages: pypandoc
Successfully installed pypandoc-1.10
Mounted at /content/drive
cp: cannot stat 'drive/My Drive/Colab Notebooks/FinalProject.ipynb': No such
file or directory
[NbConvertApp] WARNING | pattern 'FinalProject.ipynb' matched no files
This application is used to convert notebook files (*.ipynb)
        to various other formats.

        WARNING: THE COMMANDLINE INTERFACE MAY CHANGE IN FUTURE RELEASES.

Options
=======
The options below are convenience aliases to configurable class-options,
as listed in the "Equivalent to" description-line of the aliases.
To see all configurable class-options for some <cmd>, use:
    <cmd> --help-all

--debug
    set log level to logging.DEBUG (maximize logging output)
    Equivalent to: [--Application.log_level=10]
--show-config
    Show the application's configuration (human-readable format)
    Equivalent to: [--Application.show_config=True]
--show-config-json
    Show the application's configuration (json format)
    Equivalent to: [--Application.show_config_json=True]
--generate-config
    generate default config file
    Equivalent to: [--JupyterApp.generate_config=True]
-y
    Answer yes to any questions instead of prompting.
    Equivalent to: [--JupyterApp.answer_yes=True]
--execute
    Execute the notebook prior to export.
    Equivalent to: [--ExecutePreprocessor.enabled=True]
--allow-errors
    Continue notebook execution even if one of the cells throws an error and
include the error message in the cell output (the default behaviour is to abort
conversion). This flag is only relevant if '--execute' was specified, too.
    Equivalent to: [--ExecutePreprocessor.allow_errors=True]
--stdin
    read a single notebook file from stdin. Write the resulting notebook with
default basename 'notebook.*'
    Equivalent to: [--NbConvertApp.from_stdin=True]
```

```
--stdout
    Write notebook output to stdout instead of files.
    Equivalent to: [--NbConvertApp.writer_class=StdoutWriter]
--inplace
    Run nbconvert in place, overwriting the existing notebook (only
            relevant when converting to notebook format)
    Equivalent to: [--NbConvertApp.use_output_suffix=False
--NbConvertApp.export_format=notebook --FilesWriter.build_directory=]
--clear-output
    Clear output of current file and save in place,
            overwriting the existing notebook.
    Equivalent to: [--NbConvertApp.use_output_suffix=False
--NbConvertApp.export_format=notebook --FilesWriter.build_directory=
--ClearOutputPreprocessor.enabled=True]
--no-prompt
    Exclude input and output prompts from converted document.
    Equivalent to: [--TemplateExporter.exclude_input_prompt=True
--TemplateExporter.exclude_output_prompt=True]
--no-input
    Exclude input cells and output prompts from converted document.
            This mode is ideal for generating code-free reports.
    Equivalent to: [--TemplateExporter.exclude_output_prompt=True
--TemplateExporter.exclude_input=True]
--log-level=<Enum>
    Set the log level by value or name.
    Choices: any of [0, 10, 20, 30, 40, 50, 'DEBUG', 'INFO', 'WARN', 'ERROR',
'CRITICAL']
    Default: 30
    Equivalent to: [--Application.log_level]
--config=<Unicode>
    Full path of a config file.
    Default: ''
    Equivalent to: [--JupyterApp.config_file]
--to=<Unicode>
    The export format to be used, either one of the built-in formats
            ['asciidoc', 'custom', 'html', 'latex', 'markdown', 'notebook',
'pdf', 'python', 'rst', 'script', 'slides']
            or a dotted object name that represents the import path for an
            `Exporter` class
    Default: 'html'
    Equivalent to: [--NbConvertApp.export_format]
--template=<Unicode>
    Name of the template file to use
    Default: ''
    Equivalent to: [--TemplateExporter.template_file]
--writer=<DottedObjectName>
    Writer class used to write the
                                        results of the conversion
    Default: 'FilesWriter'
    Equivalent to: [--NbConvertApp.writer_class]
--post=<DottedOrNone>
    PostProcessor class used to write the
                                    results of the conversion
    Default: ''
    Equivalent to: [--NbConvertApp.postprocessor_class]
--output=<Unicode>
    overwrite base name use for output files.
                can only be used when converting one notebook at a time.
    Default: ''
    Equivalent to: [--NbConvertApp.output_base]
--output-dir=<Unicode>
    Directory to write output(s) to. Defaults
                                  to output to the directory of each notebook.
To recover
                                  previous default behaviour (outputting to the
current
                                  working directory) use . as the flag value.
    Default: ''
    Equivalent to: [--FilesWriter.build_directory]
--reveal-prefix=<Unicode>
    The URL prefix for reveal.js (version 3.x).
            This defaults to the reveal CDN, but can be any url pointing to a
copy
            of reveal.js.
            For speaker notes to work, this must be a relative path to a local
            copy of reveal.js: e.g., "reveal.js".
            If a relative path is given, it must be a subdirectory of the
            current directory (from which the server is run).
            See the usage documentation
            (https://nbconvert.readthedocs.io/en/latest/usage.html#reveal-js-
html-slideshow)
            for more details.
    Default: ''
    Equivalent to: [--SlidesExporter.reveal_url_prefix]
--nbformat=<Enum>
    The nbformat version to write.
            Use this to downgrade notebooks.
    Choices: any of [1, 2, 3, 4]
    Default: 4
    Equivalent to: [--NotebookExporter.nbformat_version]

Examples
--------

    The simplest way to use nbconvert is
```

```
            > jupyter nbconvert mynotebook.ipynb

            which will convert mynotebook.ipynb to the default format (probably
HTML).

            You can specify the export format with `--to`.
            Options include ['asciidoc', 'custom', 'html', 'latex', 'markdown',
'notebook', 'pdf', 'python', 'rst', 'script', 'slides'].

            > jupyter nbconvert --to latex mynotebook.ipynb

            Both HTML and LaTeX support multiple output templates. LaTeX
includes
            'base', 'article' and 'report'.  HTML includes 'basic' and 'full'.
You
            can specify the flavor of the format used.

            > jupyter nbconvert --to html --template basic mynotebook.ipynb

            You can also pipe the output to stdout, rather than a file

            > jupyter nbconvert mynotebook.ipynb --stdout

            PDF is generated via latex

            > jupyter nbconvert mynotebook.ipynb --to pdf

            You can get (and serve) a Reveal.js-powered slideshow

            > jupyter nbconvert myslides.ipynb --to slides --post serve

            Multiple notebooks can be given at the command line in a couple of
            different ways:

            > jupyter nbconvert notebook*.ipynb
            > jupyter nbconvert notebook1.ipynb notebook2.ipynb

            or you can specify the notebooks list in a config file, containing::

                c.NbConvertApp.notebooks = ["my_notebook.ipynb"]

            > jupyter nbconvert --config mycfg.py

To see all available configurables, use `--help-all`.
```