

# Module 19 Challenge

Start Assignment

- Due Nov 7 by 11:59pm
- Points 100
- Submitting a text entry box or a website url
- Attempts 0
- Allowed Attempts 4

In this challenge, you'll use your knowledge of Python and unsupervised learning to predict if cryptocurrencies are affected by 24-hour or 7-day price changes.

## Before You Begin

1. Create a new repository for this project called `CryptoClustering`. **Do not add this homework to an existing repository.**
2. Clone the new repository to your computer.
3. Push your changes to GitHub.

## Files

Download the following files to help you get started:

**Module 19 Challenge files** ➔ [https://static.bc-edx.com/data/dl-1-2/m19/lms/starter/Starter\\_Code.zip](https://static.bc-edx.com/data/dl-1-2/m19/lms/starter/Starter_Code.zip)

## Instructions

1. Rename the `Crypto_Clustering_starter_code.ipynb` file as `Crypto_Clustering.ipynb`.
2. Load the `crypto_market_data.csv` into a DataFrame.
3. Get the summary statistics and plot the data to see what the data looks like before proceeding.

## Prepare the Data

- Use the `StandardScaler()` module from `scikit-learn` to normalize the data from the CSV file.
- Create a DataFrame with the scaled data and set the "coin\_id" index from the original DataFrame as the index for the new DataFrame.
  - The first five rows of the scaled DataFrame should appear as follows:

	price_change_percentage_24h	price_change_percentage_7d	price_change_percentage_14d	price_change_percentage_30d	price_change_percentage_60d
coin_id					
bitcoin	0.508529	0.493193	0.772200	0.235460	-0.067495
ethereum	0.185446	0.934445	0.558692	-0.054341	-0.273483
tether	0.021774	-0.706337	-0.021680	-0.061030	0.008005
ripple	-0.040764	-0.810928	0.249458	-0.050388	-0.373164
bitcoin-cash	1.193036	2.000959	1.760610	0.545842	-0.291203

## Find the Best Value for k Using the Scaled DataFrame

Use the elbow method to find the best value for `k` using the following steps:

- Create a list with the number of k values from 1 to 11.
- Create an empty list to store the inertia values.
- Create a `for` loop to compute the inertia with each possible value of `k`.
- Create a dictionary with the data to plot the elbow curve.
- Plot a line chart with all the inertia values computed with the different values of `k` to visually identify the optimal value for `k`.
- Answer the following question in your notebook: What is the best value for `k`?

## Cluster Cryptocurrencies with K-means Using the Scaled DataFrame

Use the following steps to cluster the cryptocurrencies for the best value for `k` on the scaled DataFrame:

- Initialize the K-means model with the best value for `k`.
- Fit the K-means model using the scaled DataFrame.
- Predict the clusters to group the cryptocurrencies using the scaled DataFrame.
- Create a copy of the scaled DataFrame and add a new column with the predicted clusters.
- Create a scatter plot using hvPlot as follows:
  - Set the x-axis as "price\_change\_percentage\_24h" and the y-axis as "price\_change\_percentage\_7d".
  - Color the graph points with the labels found using K-means.
  - Add the "coin\_id" column in the `hover_cols` parameter to identify the cryptocurrency represented by each data point.

## Optimize Clusters with Principal Component Analysis

- Using the original scaled DataFrame, perform a PCA and reduce the features to three principal components.
- Retrieve the explained variance to determine how much information can be attributed to each principal component and then answer the following question in your notebook:
  - What is the total explained variance of the three principal components?
- Create a new DataFrame with the scaled PCA data and set the "coin\_id" index from the original DataFrame as the index for the new DataFrame.

- The first five rows of the scaled PCA DataFrame should appear as follows:

	PC1	PC2	PC3
coin_id			
bitcoin	-0.600667	0.842760	0.461595
ethereum	-0.458261	0.458466	0.952877
tether	-0.433070	-0.168126	-0.641752
ripple	-0.471835	-0.222660	-0.479053
bitcoin-cash	-1.157800	2.041209	1.859715

### Find the Best Value for k Using the PCA DataFrame

Use the elbow method on the scaled PCA DataFrame to find the best value for `k` using the following steps:

- Create a list with the number of k-values from 1 to 11.
- Create an empty list to store the inertia values.
- Create a `for` loop to compute the inertia with each possible value of `k`.
- Create a dictionary with the data to plot the Elbow curve.
- Plot a line chart with all the inertia values computed with the different values of `k` to visually identify the optimal value for `k`.
- Answer the following question in your notebook:
  - What is the best value for `k` when using the scaled PCA DataFrame?
  - Does it differ from the best k value found using the original scaled DataFrame?

### Cluster Cryptocurrencies with K-means Using the PCA DataFrame


Use the following steps to cluster the cryptocurrencies for the best value for `k` on the PCA DataFrame:

- Initialize the K-means model with the best value for `k`.
- Fit the K-means model using the scaled PCA DataFrame.

- Predict the clusters to group the cryptocurrencies using the scaled PCA DataFrame.
- Create a copy of the scaled PCA DataFrame and add a new column to store the predicted clusters.
- Create a scatter plot using hvPlot as follows:
  - Set the x-axis as "PC1" and the y-axis as "PC2".
  - Color the graph points with the labels found using K-means.
  - Add the "coin\_id" column in the `hover_cols` parameter to identify the cryptocurrency represented by each data point.
- Answer the following question:
  - What is the impact of using fewer features to cluster the data using K-Means?



## REWIND

Recall that you learned how to create composite plots in a previous module. If you need a refresher on how to create these plots, review that module. You can also check [Composing Plots](https://holoviz.org/tutorial/Composing_Plots.html)  ([https://holoviz.org/tutorial/Composing\\_Plots.html](https://holoviz.org/tutorial/Composing_Plots.html)) in the hvPlot documentation.

## Requirements

### Find the Best Value for k by Using the Scaled DataFrame (15 points)

To receive all points, you must:

- Code the elbow method algorithm to find the best value for k. Use a range from 1 to 11. (5 points)
- To visually identify the optimal value for k, plot a line chart of all the inertia values computed with the different values of k. (5 points)
- Answer the following question: What's the best value for k? (5 points)

### Cluster the Cryptocurrencies with K-Means by Using the Scaled DataFrame (10 points)

To receive all points, you must:

- Initialize the K-means model with best value for k. (1 point)
- Fit the K-means model by using the scaled DataFrame. (1 point)
- Predict the clusters for grouping the cryptocurrencies by using the scaled DataFrame. Review the resulting array of cluster values. (3 points)
- Create a copy of the scaled DataFrame, and then add a new column of the predicted clusters. (1 point)

- Using hvPlot, create a scatter plot by setting `x="price_change_percentage_24h"` and `y="price_change_percentage_7d"`. Color the graph points with the labels that you found by using K-means. Then add the crypto name to the `hover_cols` parameter to identify the cryptocurrency that each data point represents. (4 points)

## Optimize the Clusters with Principal Component Analysis (10 points)

To receive all points, you must:

- Create a PCA model instance, and set `n_components=3`. (1 point)
- Use the PCA model to reduce the features to three principal components. (2 points)
- Get the explained variance to determine how much information can be attributed to each principal component. (2 points)
- Answer the following question: What's the total explained variance of the three principal components? (3 points)
- Create a new DataFrame from the scaled PCA data. Be sure to set the `coin_id` index from the original scaled DataFrame as the index for the new DataFrame. Review the first five rows of the DataFrame. (2 points)

## Find the Best Value for k by Using the PCA DataFrame (10 points)

To receive all points, you must:

- Code the elbow method algorithm, and use the scaled PCA DataFrame to find the best value for k. Use a range from 1 to 11. (2 points)
- To visually identify the optimal value for k, plot a line chart of all the inertia values computed with the different values of k. (5 points)
- Answer the following questions: What's the best value for k when using the scaled PCA DataFrame? Does it differ from the best value for k that you found by using the original scaled DataFrame? (3 points)

## Cluster the Cryptocurrencies with K-means by Using the PCA DataFrame (10 points)

To receive all points, you must:

- Initialize the K-means model with the best value for k. (1 point)
- Fit the K-means model by using the PCA data. (1 point)
- Predict the clusters for grouping the cryptocurrencies by using the PCA data. Review the resulting array of cluster values. (3 points)
- Create a copy of the scaled PCA DataFrame, and then add a new column of the predicted clusters. (1 point)
- Using hvPlot, create a scatter plot by setting `x="PC1"` and `y="PC2"`. Color the graph points with the labels that you found by using K-means. Then add the crypto name to the `hover_cols` parameter to identify the cryptocurrency that each data point represents. (4 points)

## Visualize and Compare the Results (15 points)

To receive all points, you must:

- Create a composite plot by using hvPlot and the plus sign ( $\oplus$ ) operator to compare the elbow curve that you created from the original scaled DataFrame with the one that you created from the scaled PCA DataFrame. (5 points)
- Create a composite plot by using hvPlot and the plus ( $\oplus$ ) operator to compare the cryptocurrency clusters that resulted from using the original scaled DataFrame with those that resulted from the scaled PCA DataFrame. (5 points)
- Answer the following question: Based on visually analyzing the cluster analysis results, what's the impact of using fewer features to cluster the data by using K-means? (5 points)

## Coding Conventions and Formatting (10 points)

To receive all points, you must:

- Place imports at the top of the file, just after any module comments and docstrings, and before module globals and constants. (3 points)
- Name functions and variables with lowercase characters, with words separated by underscores. (2 points)
- Follow DRY (Don't Repeat Yourself) principles, creating maintainable and reusable code. (3 points)
- Use concise logic and creative engineering where possible. (2 points)

## Deployment and Submission (10 points)

To receive all points, you must:

- Submit a link to a GitHub repository that's cloned to your local machine and that contains your files. (4 points)
- Use the command line to add your files to the repository. (3 points)
- Include appropriate commit messages in your files. (3 points)

## Code Comments (10 points)

To receive all points, your code must:

- Be well commented with concise, relevant notes that other developers can understand. (10 points)

## Grading

This project will be evaluated against the requirements and assigned a grade according to the following table:

Grade	Points
A (+/-)	90+
B (+/-)	80–89
C (+/-)	70–79

Grade	Points
D (+/-)	60–69
F (+/-)	< 60

## Submission

To submit your Challenge assignment, click Submit, and then provide the URL of your GitHub repository for grading.

### NOTE


You are allowed to miss up to two Challenge assignments and still earn your certificate. If you complete all Challenge assignments, your lowest two grades will be dropped. If you wish to skip this assignment, click Next, and move on to the next module.

Comments are disabled for graded submissions in Bootcamp Spot. If you have questions about your feedback, please notify your instructional staff or your Student Success Advisor. If you would like to resubmit your work for an additional review, you can use the Resubmit Assignment button to upload new links. You may resubmit up to three times for a total of four submissions.

### IMPORTANT

**It is your responsibility to include a note in the README section of your repo specifying code source and its location within your repo.** This applies if you have worked with a peer on an assignment, used code in which you did not author or create sourced from a forum such as Stack Overflow, or you received code outside curriculum content from support staff such as an Instructor, TA, Tutor, or Learning Assistant. This will provide visibility to grading staff of your circumstance in order to avoid flagging your work as plagiarized.

If you are struggling with a challenge assignment or any aspect of the academic curriculum, please remember that there are student support services available for you:

1. Ask the class Slack channel/peer support.
2. AskBCS Learning Assistants exists in your class Slack application.
3. Office hours facilitated by your instructional staff before and after each class session.
4. **[Tutoring Guidelines](https://docs.google.com/document/d/1hTIdEfWhX21B_Vz9ZentkPeziu4pPfnwiZbwQB27E90/edit?usp=sharing)**   
([https://docs.google.com/document/d/1hTIdEfWhX21B\\_Vz9ZentkPeziu4pPfnwiZbwQB27E90/edit?usp=sharing](https://docs.google.com/document/d/1hTIdEfWhX21B_Vz9ZentkPeziu4pPfnwiZbwQB27E90/edit?usp=sharing))-  
schedule a tutor session in the Tutor Sessions section of Bootcampspot - Canvas
5. If the above resources are not applicable and you have a need, please reach out to a member of your instructional team, your Student Success Advisor, or submit a support ticket in the Student Support section of your BCS application.

## References

Data for this dataset was generated by edX Boot Camps LLC, and is intended for educational purposes only.

© 2024 edX Boot Camps LLC