

1. Unit 13 Homework Assignment - The Power of the Cloud and Unsupervised Learning

1. Background

1. Before You Begin

2. Option 1: Robo Advisor for Retirement Plans

1. Background

2. Files

3. Instructions

1. Initial Robo Advisor Configuration

2. Build and Test the Robo Advisor

3. Enhance the Robo Advisor with an Amazon Lambda Function

1. User Input Validation

2. Investment Portfolio Recommendation

4. Submission

5. Hints

3. Option 2: Clustering Crypto

1. Background

2. Files

3. Instructions

1. Data Preprocessing

2. Reducing Data Dimensions Using PCA

3. Clustering Cryptocurrencies Using K-Means

4. Visualizing Results

4. Optional Challenge

1. Complementary Resources

5. Submission

Unit 13 Homework Assignment - The Power of the Cloud and Unsupervised Learning

Background

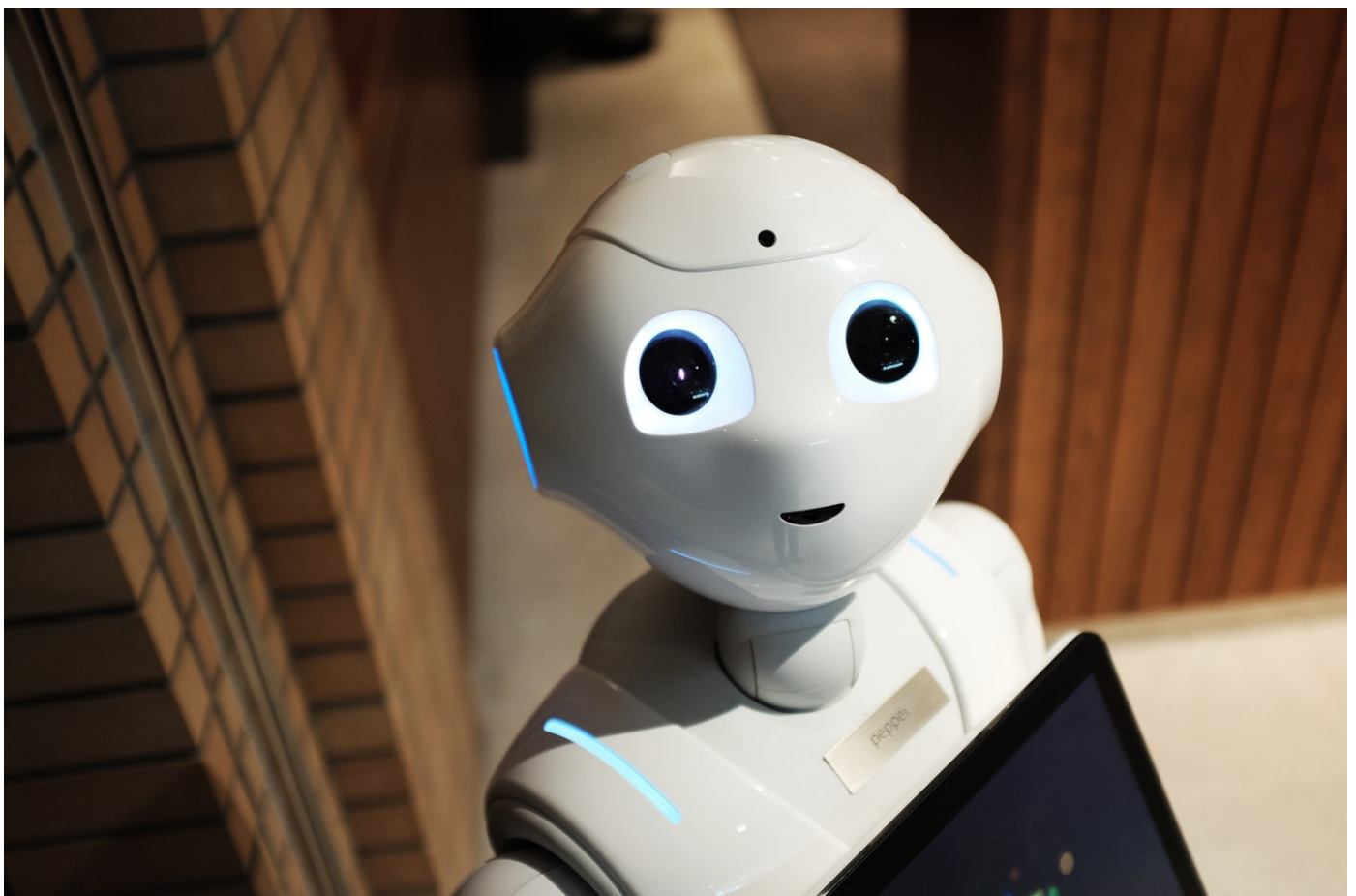
It is time to take what you have learned about unsupervised learning and the AWS services and apply it to new situations. For this assignment, you will need to complete

one of two (not both) challenges. Which challenge you take on is your choice. Just be sure to give it your all -- as the skills you hone will become powerful tools in your FinTech tool belt.

Before You Begin

1. Create a new repository for this project called **unit13-challenge**. **Do not add this homework to an existing repository.**
2. Clone the new repository to your computer.
3. Inside your local git repository, create a directory for the challenge assignment you choose. Use folder names corresponding to the challenges: **RoboAdvisor** or **ClusteringCrypto**.
4. Add your solution files to this folder.
5. Push the above changes to GitHub or GitLab.

Option 1: Robo Advisor for Retirement Plans



Background

You were hired as a digital transformation consultant by one of the most prominent retirement plan providers in the country; they want to increase their client portfolio, especially by engaging young people. Since machine learning and NLP are disrupting finance to improve customer experience, you decide to create a robo advisor that could be used by customers or potential new customers to get investment portfolio recommendations for retirement.

In this homework assignment, you will combine your new Amazon Web Services skills with your already mastered Python superpowers, to create a bot that will recommend an investment portfolio for a retirement plan.

You are asked to accomplish the following main tasks:

1. **Initial Robo Advisor Configuration:** Define an Amazon Lex bot with a single intent that establishes a conversation about the requirements to suggest an investment portfolio for retirement.
2. **Build and Test the Robo Advisor:** Make sure that your bot is working and responding accurately along with the conversation with the user, by building and testing it.
3. **Enhance the Robo Advisor with an Amazon Lambda Function:** Create an Amazon Lambda function that validates the user's input and returns the investment portfolio recommendation. This task includes testing the Amazon Lambda function and making the integration with the bot.

Files

- [lambda_function.py](#)
 - [correct_dialog.txt](#)
 - [age_error.txt](#)
 - [incorrect_amount_error.txt](#)
 - [negative_age_error.txt](#)
-

Instructions

Initial Robo Advisor Configuration

In this section, you will create the **RoboAdvisor** bot and add an intent with its corresponding slots.

Sign in into your AWS Management Console and [create a new custom Amazon Lex bot](#). Use the following parameters:

- **Bot name:** RoboAdvisor
- **Output voice:** Salli
- **Session timeout:** 5 minutes
- **Sentiment analysis:** No
- **COPPA:** No
- **Advanced options:** No
- *Leave default values for all other options.*

Create the **RecommendPortfolio** intent, and configure some sample utterances as follows (you can add more utterances as you wish):

- I want to save money for my retirement
- I'm {age} and I would like to invest for my retirement
- I'm {age} and I want to invest for my retirement
- I want the best option to invest for my retirement
- I'm worried about my retirement
- I want to invest for my retirement
- I would like to invest for my retirement

This bot will use four slots, three using built-in types and one custom slot named **riskLevel**. Define the three initial slots as follows:

Name	Slot Type	Prompt
firstName	AMAZON.US_FIRST_NAME	Thank you for trusting me to help, could you please give me your name?
age	AMAZON.NUMBER	How old are you?
investmentAmount	AMAZON.NUMBER	How much do you want to invest?

The **riskLevel** custom slot will be used to retrieve the risk level the user is willing to take on the investment portfolio. Create this custom slot as follows:

- Select the **+** icon next to 'Slot Types' in the 'Editor' on the left side of the screen.
- Choose **create custom slot** from the resulting display window.
- For **Slot type name**, type: riskLevel
- Select the radial dial button next to **Restrict to Slot values and synonyms**, then fill in the appropriate values and synonyms. *Example:* Low, Minimal; High, Maximum.
- Click **Add slot to intent** when finished.

To format the response cards for the intent, click on the gear icon next to the intent as seen in the image below:

Slots ⓘ						
Priority	Required	Name	Slot type	Version	Prompt	Settings
		e.g. Location	e.g. AMAZON.US_CI...		e.g. What city?	+
1.	✓	firstName	AMAZON.US_FIRST...	Built-in	for trusting me t...	⚙️
2.	✓	age	AMAZON.NUMBER	Built-in	How old are you?	⚙️
3.	✓	investmentAmount	AMAZON.NUMBER	Built-in	How much do you want to invest...	⚙️
4.	✓	riskLevel	ariskLevel	Latest	e.g. What city?	⚙️

Next, input the following data in the resulting display window:

- **Prompt:** What level of investment risk would you like to take?
- **Maximum number of retries:** 2
- **Prompt response cards:** 4

Configure the response cards for the **riskLevel** slot as is shown bellow:


Card 1	Card 2
--------	--------

Card 1

Card 1 ⓘ

Preview as: Facebook ▾

🗑️


[Edit Image Url](#)

Title*

None

Subtitle*

No risk at all

Button title*

None

Button value*


None ▾

Card 2

Card 2 ⓘ

Preview as: Facebook ▾

🗑️


[Edit Image Url](#)

Title*

Very Low or Low

Subtitle*

Just a bit of risk

Button title*

Very Low

✕

Button value*

Very Low ▾

Button title

Low

✕

Button value


Low ▾

Card 3

Card 3 ⓘ

Preview as: Facebook ▾

🗑️


[Edit Image Url](#)

Title*

Medium

Subtitle*

Let's start becoming wild

Button title*

Medium

Button value*


Medium ▾

Card 4

Card 4 ⓘ

Preview as: Facebook ▾

🗑️


[Edit Image Url](#)

Title*

High or Very High

Subtitle*

I have no fear!

Button title*

High

✕

Button value*

High ▾

Button title

Very High

✕

Button value

Very High ▾

Note: You can download free icons from [this website](#) or you can use the icons provided in the [Icons directory](#).

Move to the *Confirmation Prompt* section, and set the following messages:

- **Confirm:** Thanks, now I will look for the best investment portfolio for you.
- **Cancel:** I will be pleased to assist you in the future.

Leave the error handling configuration for the [RecommendPortfolio](#) bot with the default values.

< RoboAdvisor Latest ▾

Editor

Settings

Channels

Monitoring

Intents

RecommendPortfolio

Slot types

RiskLevel

Error Handling

Error handling

Clarification prompts

e.g. Sorry, can you please repeat that?

Sorry, can you please repeat that?

Maximum number of retries

2

Hang-up phrase

e.g. Sorry, I could not understand. Please contact customer support.

Sorry, I could not understand. Goodbye.

Save

Build and Test the Robo Advisor

In this section, you will test your Robo Advisor. To build your bot, click on the **Build** button in the upper right hand corner. Once the build is complete, test it in the chatbot window. You should see a conversation like the one below.

You're now ready for testing. Type an utterance below to begin conversation with your chatbot.

Clear chat history



Chat with your bot...

Enhance the Robo Advisor with an Amazon Lambda Function

In this section, you will create an Amazon Lambda function that will validate the data provided by the user on the Robo Advisor. Start by creating a new lambda function from scratch and name it `recommendPortfolio`. Select Python 3.7 as runtime.

In the Lambda function, start by deleting the AWS generated default lines of code, then paste in the starter code provided in [lambda_function.py](#) and complete the `recommend_portfolio()` function by following these guidelines:

User Input Validation

- The `age` should be greater than zero and less than 65.
- the `investment_amount` should be equal to or greater than 5000.

Investment Portfolio Recommendation

Once the intent is fulfilled, the bot should response with an investment recommendation based on the selected risk level as follows:

- **none:** "100% bonds (AGG), 0% equities (SPY)"
- **very low:** "80% bonds (AGG), 20% equities (SPY)"
- **low:** "60% bonds (AGG), 40% equities (SPY)"
- **medium:** "40% bonds (AGG), 60% equities (SPY)"
- **high:** "20% bonds (AGG), 80% equities (SPY)"
- **very high:** "0% bonds (AGG), 100% equities (SPY)"

Be creative while coding your solution, you can have all the code on the `recommend_portfolio()` function, or you can split the functionality across different functions, put your Python coding skills in action!

Once you finish coding your lambda function, test it using the [sample test cases](#) provided for this homework.

After successfully testing your code, open the Amazon Lex Console and navigate to the `RecommendPortfolio` bot configuration, integrate your new lambda function by selecting it in the *Lambda initialization and validation* and *Fulfillment* sections. Build your bot, and you should have a conversation as follows.

> Test bot (Latest)

✓ Ready. Build complete.

You're now ready for testing. Type an utterance below to begin conversation with your chatbot.

Clear chat history



Chat with your bot...

Submission

You should create a brand new repository in GitHub and upload the following files to your repo.

- A python script with your final lambda function.
- From the Amazon Lex Console, export your bot, intent, and slot using **Amazon Lex** as the target platform, and upload the ZIP files to your repo.
- Create a short video or animated GIF showing a demo of your Robo Advisor in action from the test window. Upload the video or animated GIF file to your repo.

Once you have uploaded all the files into the repo, post a link to your homework's repository in BootCamp Spot.

Hints

- Make sure your intent and slot names are named correctly in your Lambda code. The names in Lex are supposed to match the names in Lambda exactly:

The image shows two screenshots from the AWS console. The left screenshot displays the Lambda function code for 'lambda_function'. The right screenshot shows the Amazon Lex console for the 'RoboAdvisor_test' intent.

Left Screenshot (Lambda Function Code):

```
170 output_session_attributes = intent_request["sessionAttributes"]
171
172 return delegate(output_session_attributes, get_slots(intent_request))
173
174 # Get the initial investment recommendation
175 initial_recommendation = get_investment_recommendation(risk_level)
176
177 # Return a message with the initial recommendation based on the risk level.
178 return close(
179     intent_request["sessionAttributes"],
180     "Fulfilled",
181     {
182         "contentType": "PlainText",
183         "content": ""() thank you for your information;
184         based on the risk level you defined, my recommendation is to choose an investment
185         """
186         first_name, initial_recommendation
187     },
188 ),
189 )
190
191
192 ## Intents Dispatcher ##
193 def dispatch(intent_request):
194     """
195     Called when the user specifies an intent for this bot.
196     """
197     intent_name = intent_request["currentIntent"]["name"]
198
199     # Dispatch to bot's intent handlers
200     if intent_name == "RecommendPortfolio":
201         return recommend_portfolio(intent_request)
202
203     raise Exception("Intent with name " + intent_name + " not supported")
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
```

Right Screenshot (Amazon Lex Console):

The console shows the 'RecommendPortfolio_test' intent. The 'Sample utterances' section lists several phrases, with 'age' highlighted as a slot. The 'Slots' section shows a table of slots:

Priority	RequiredName	Slot type	Version	Prompt	Settings
1.	firstName	AMAZ...	Built-in	Thank you for trus	⚙️
2.	age	AMAZ...	Built-in	How old are you?	⚙️
3.	investmentAr	AMAZ...	Built-in	How much do you	⚙️
4.	riskLevel	riskLev...	1	What level of inve	⚙️

- You may have to refresh the Lex intent page after creating the custom slot and the lambda function in order to see them in the options.
- If you are using a Mac, you can create a screen-recording using the built-in QuickTime player. Follow [this link](#) to learn more.
- If you are using Windows 10, you can create a screen-recording using the built-in Xbox Game Bar. Follow [this link](#) to learn more.

Option 2: Clustering Crypto



Cryptocurrencies coins by Worldspectrum | Free License

Background

You are a Senior Manager at the Advisory Services team on a [Big Four firm](#). One of your most important clients, a prominent investment bank, is interested in offering a new cryptocurrencies investment portfolio for its customers, however, they are lost in the immense universe of cryptocurrencies. They ask you to help them make sense of it all by generating a report of what cryptocurrencies are available on the trading market and how they can be grouped using classification.

In this homework assignment, you will put your new unsupervised learning and Amazon SageMaker skills into action by clustering cryptocurrencies and creating plots to present your results.

You are asked to accomplish the following main tasks:

- **Data Preprocessing:** Prepare data for dimension reduction with PCA and clustering using K-Means.

- **Reducing Data Dimensions Using PCA:** Reduce data dimension using the `PCA` algorithm from `sklearn`.
 - **Clustering Cryptocurrencies Using K-Means:** Predict clusters using the cryptocurrencies data using the `KMeans` algorithm from `sklearn`.
 - **Visualizing Results:** Create some plots and data tables to present your results.
 - **Optional Challenge:** Deploy your notebook to Amazon SageMaker.
-

Files

- [crypto_clustering.ipynb](#)
-

Instructions

Data Preprocessing

In this section, you will load the information about cryptocurrencies and perform data preprocessing tasks. You can choose one of the following methods to load the data:

1. Using the provided `CSV` file, create a `Path` object and read the file data directly into a DataFrame named `crypto_df` using `pd.read_csv()`.
2. Using the following `requests` library, retrieve the necessary data from the following API endpoint from *CryptoCompare* - `https://min-api.cryptocompare.com/data/all/coinlist`. **HINT:** You will need to use the 'Data' key from the json response, then transpose the DataFrame. Name your DataFrame `crypto_df`.

With the data loaded into a Pandas DataFrame, continue with the following data preprocessing tasks.

3. Keep only the necessary columns:
'CoinName', 'Algorithm', 'IsTrading', 'ProofType', 'TotalCoinsMined', 'TotalCoinSupply'
4. Keep only the cryptocurrencies that are trading.
5. Keep only the cryptocurrencies with a working algorithm.
6. Remove the `IsTrading` column.

7. Remove all cryptocurrencies with at least one null value.
8. Remove all cryptocurrencies that have no coins mined.
9. Drop all rows where there are 'N/A' text values.
10. Store the names of all cryptocurrencies in a DataFrame named `coins_name`, use the `crypto_df.index` as the index for this new DataFrame.
11. Remove the `CoinName` column.
12. Create dummy variables for all the text features, and store the resulting data in a DataFrame named `X`.
13. Use the `StandardScaler` from `sklearn` to standardize all the data of the `X` DataFrame. Remember, this is important prior to using PCA and K-Means algorithms.

Reducing Data Dimensions Using PCA

Use the `PCA` algorithm from `sklearn` to reduce the dimensions of the `X` DataFrame down to three principal components.

Once you have reduced the data dimensions, create a DataFrame named `pcs_df` using as columns names "PC 1", "PC 2" and "PC 3"; use the `crypto_df.index` as the index for this new DataFrame.

You should have a DataFrame like the following:

	PC 1	PC 2	PC 3
42	-0.335587	1.060721	-0.442783
404	-0.319468	1.060956	-0.443089
1337	2.298643	1.616176	-0.543968
BTC	-0.146783	-1.352007	0.158842
ETH	-0.134787	-1.929485	0.248370
LTC	-0.157018	-1.083072	0.028331
DASH	-0.409440	1.226179	-0.562194
XMR	-0.165529	-2.462011	0.404730
ETC	-0.133224	-1.929573	0.248345
ZEC	-0.128082	-1.987514	0.351649

Clustering Cryptocurrencies Using K-Means

In this section, you will use the **KMeans** algorithm from **sklearn** to cluster the cryptocurrencies using the PCA data.

Perform the following tasks:

1. Create an Elbow Curve to find the best value for **k** using the **pcs_df** DataFrame.
2. Once you define the best value for **k**, run the **Kmeans** algorithm to predict the **k** clusters for the cryptocurrencies data. Use the **pcs_df** to run the **KMeans** algorithm.
3. Create a new DataFrame named **clustered_df**, that includes the following columns **"Algorithm"**, **"ProofType"**, **"TotalCoinsMined"**, **"TotalCoinSupply"**, **"PC 1"**, **"PC 2"**, **"PC 3"**, **"CoinName"**, **"Class"**. You should maintain the index of the **crypto_df** DataFrames as is shown bellow.

	Algorithm	ProofType	TotalCoinsMined	TotalCoinSupply	PC 1	PC 2	PC 3	CoinName	Class
42	Scrypt	PoW/PoS	42	42	-0.335587	1.060721	-0.442783	42 Coin	0
404	Scrypt	PoW/PoS	1.00862e+09	532000000	-0.319468	1.060956	-0.443089	404Coin	0
1337	X13	PoW/PoS	2.92794e+10	314159265359	2.298643	1.616176	-0.543968	EliteCoin	0
BTC	SHA-256	PoW	17918662	21000000	-0.146783	-1.352007	0.158842	Bitcoin	1
ETH	Ethash	PoW	1.07626e+08	0	-0.134787	-1.929485	0.248370	Ethereum	1
LTC	Scrypt	PoW	6.30392e+07	84000000	-0.157018	-1.083072	0.028331	Litecoin	1
DASH	X11	PoW/PoS	9.02401e+06	22000000	-0.409440	1.226179	-0.562194	Dash	0
XMR	CryptoNight-V7	PoW	1.71937e+07	0	-0.165529	-2.462011	0.404730	Monero	1
ETC	Ethash	PoW	113255332	210000000	-0.133224	-1.929573	0.248345	Ethereum Classic	1
ZEC	Equihash	PoW	7.35252e+06	21000000	-0.128082	-1.987514	0.351649	ZCash	1

Visualizing Results

In this section, you will create some data visualization to present the final results.

Perform the following tasks:

1. Create a 3D-Scatter using Plotly Express to plot the clusters using the **clustered_df** DataFrame. You should include the following parameters on the plot: **hover_name="CoinName"** and **hover_data=["Algorithm"]** to show this additional info on each data point.
2. Use **hvplot.table** to create a data table with all the current tradable cryptocurrencies. The table should have the following columns: **"CoinName"**, **"Algorithm"**, **"ProofType"**, **"TotalCoinSupply"**, **"TotalCoinsMined"**, **"Class"**
3. Create a scatter plot using **hvplot.scatter**, to present the clustered data about cryptocurrencies having **x="TotalCoinsMined"** and **y="TotalCoinSupply"** to

contrast the number of available coins versus the total number of mined coins. Use the `hover_cols=["CoinName"]` parameter to include the cryptocurrency name on each data point.

Optional Challenge

For the challenge section, you have to upload your Jupyter notebook to Amazon SageMaker and deploy it.

The `hvplot` and Plotly Express libraries are not included in the built-in anaconda environments, so for this challenge section, you should use the `altair` library instead.

Perform the following tasks:

1. Upload your Jupyter notebook and rename it as `crypto_clustering_sm.ipynb`
2. Select the `conda_python3` environment.
3. Install the `altair` library by running the following code before the initial imports.

```
!pip install -U altair
```

4. Use the `altair` scatter plot to create the Elbow Curve.
5. Use the `altair` scatter plot, instead of the 3D-Scatter from Plotly Express, to visualize the clusters. Since this is a 2D-Scatter, use `x="PC 1"` and `y="PC 2"` for the axes, and add the following columns as tool tips: `"CoinName"`, `"Algorithm"`, `"TotalCoinsMined"`, `"TotalCoinSupply"`.
6. Use the `altair` scatter plot to visualize the tradable cryptocurrencies using `x="TotalCoinsMined"` and `y="TotalCoinSupply"` for the axes.
7. Show the table of current tradable cryptocurrencies using the `display()` command.
8. Remove all `hvplot` and Plotly Express references from your code.

Complementary Resources

- [Altair visualization library website](#).
- [Simple line chart using Altair](#).

- [Simple Scatter Plot with Tooltips using Altair](#)
- [Color customization on Altair](#)
- [Printing all rows from a DataFrame](#)
- [Install External Libraries and Kernels in Amazon SageMaker Notebook Instances](#)

Submission

- Code your solution using the provided starter Jupyter notebook.
- For the *Challenge* section, create a new Jupyter notebook named `crypto_clustering_sm.ipynb` and include the necessary code to import the additional required library.
- Create and upload a repository with the above files to GitHub and post a link in BootCamp Spot.

© 2019 Trilogy Education Services, a 2U, Inc. brand. All Rights Reserved.