Data Mining in Cryptocurrency Analysis

By Group 1:

Jingyi Huang(jingyihu)

Yinghao Liu(yinghaol)

Kelly Zhang(yizhang4)

Introduction

PROBLEMS

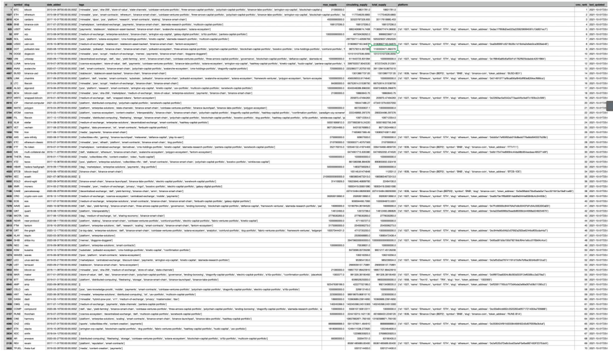
- **Prediction: Use models to check if certain cryptos are overpriced or underpriced and plan investment strategy accordingly**

- **Classification: Identify trading opportunity to see if certain cryptos deviates from its category's market behavior.**

- **Sentiment analysis & Classification: Track twitter sentiments to imply market trends for certain cryptos, market's attention and overall attitude.**

INTRODUCTION: DATA

We collected data from Coin Market Cap, which is a free database for crypto currency :https://coinmarketcap.com/historical/20211001/.
（ scrapped）
(Also in README)



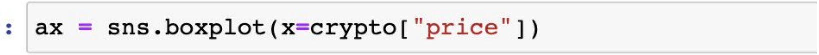
Column Name	Column Description
cmc_rank	The general ranking of popularity by Coin Market Cap
volumne_24h	The sum total of actual trades taking place.
percent_change_7d	7-day percentage of change in price
market_cap	The total value of all the coins that have been | mined.
'binance-smart-chain'	1: In BSC. 0: Not in BSC. Binance Smart Chain —BSC— is a blockchain system from the crypto-trading platform Binance.
'collectibles-nfts'	1:Digitally Collectible. 0: Not digitally collectible. (Collectibles are going digital, using a technology called non-fungible tokens (NFTs). These are digital assets — they can take forms such as photographs, music or video clips.)
'defi'	1: Decentralized. 0: Not decentralized.
'mineable'	1: Mineable. 0: Not mineable.
circulating/total_supply	Circulating supply/Total supply of that cryptocurrency
price	Price of the cryptocurrency

INTRODUCTION: DATA

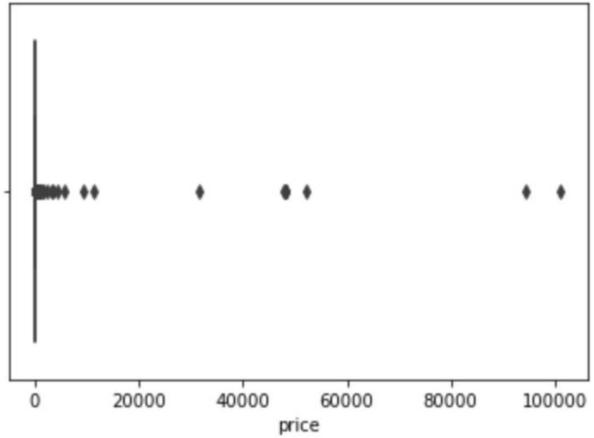
Bitcoin-related Tweets Dataset:

1. Labelled dataset for model training and validation
   - https://data.world/mercal/btc-tweets-sentiment/workspace
   - This is a tagged dataset with 3 sentiment categories: positive, neutral, and negative
2. Unlabelled dataset for real-world analysis
   - https://www.kaggle.com/kaushiksuresh147/bitcoin-tweets
   - This new dataset is unlabelled, but it contains user location information. So we would like to use the trained model to categorize the sentiment of these tweets and see how the sentiment varies across different regions.

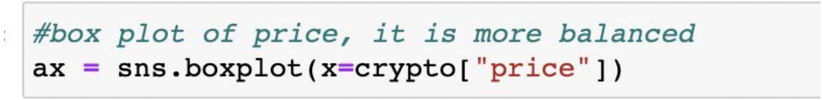
EXPLORATORY DATA ANALYSIS



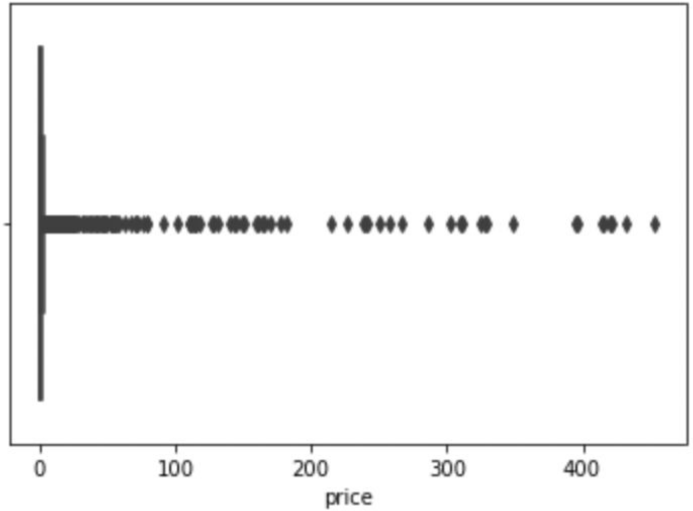
```python
ax = sns.boxplot(x=crypto["price"])
```



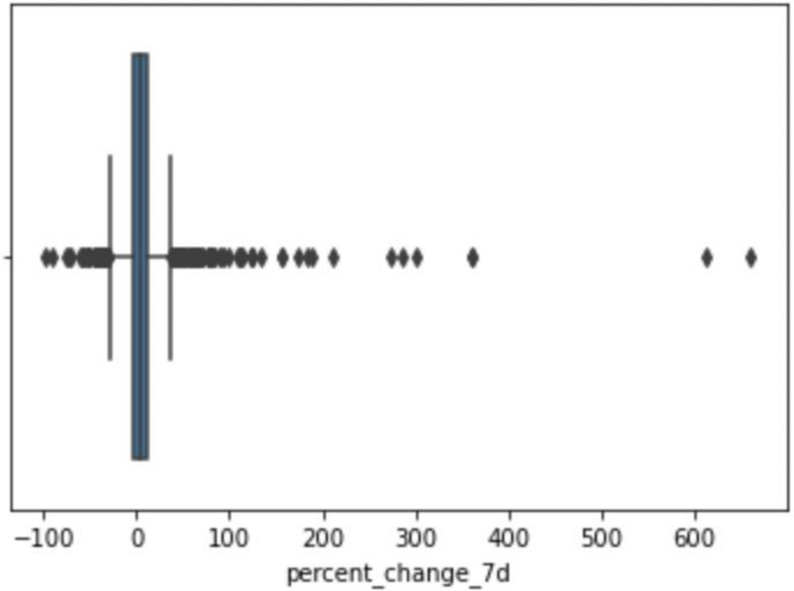
Huge range for cryptocurrencies, we decided to only deal with those that have a value less than 500.



```python
#box plot of price, it is more balanced
ax = sns.boxplot(x=crypto["price"])
```



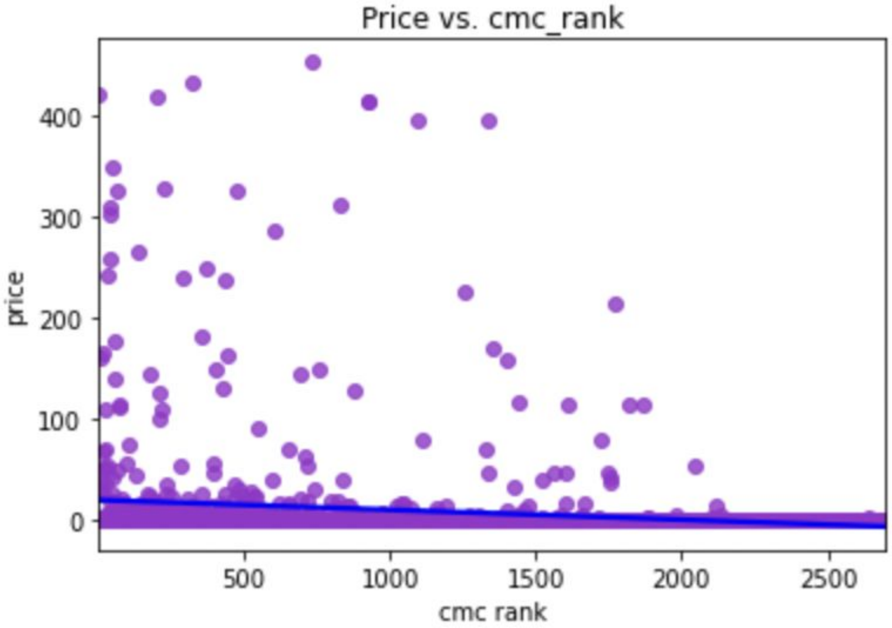
EXPLORATORY DATA ANALYSIS



We can see that many of the cryptocurrency are quite volatile, with some 7-day percentage change in price even up to 300 ~ 600%.

This is consistent with the general knowledge about cryptocurrency - having more variance than stocks. Investors of cryptocurrencies typically have high risk tolerance

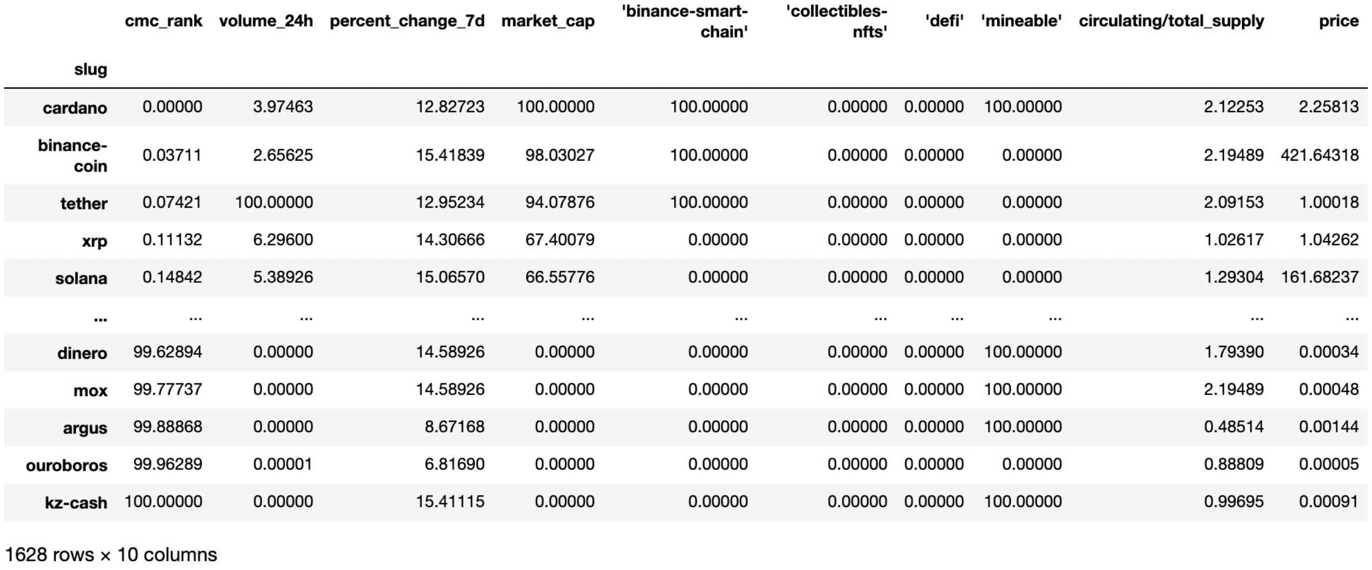
EXPLORATORY DATA ANALYSIS



- In this graph we want to explore the possible relationship between cmc rank and price.This is helpful for us to gain some prior knowledge of the variables before doing prediction and classification.

Normalization

After normalizing the data:

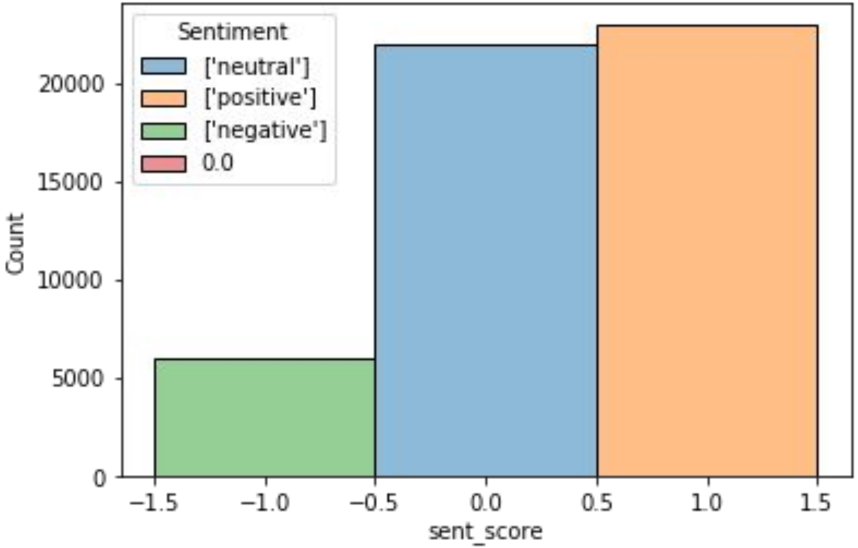


	cmc_rank	volume_24h	percent_change_7d	market_cap	'binance-smart-chain'	'collectibles-nfts'	'defi'	'mineable'	circulating/total_supply	price
slug										
cardano	0.00000	3.97463	12.82723	100.00000	100.00000	0.00000	0.00000	100.00000	2.12253	2.25813
binance-coin	0.03711	2.65625	15.41839	98.03027	100.00000	0.00000	0.00000	0.00000	2.19489	421.64318
tether	0.07421	100.00000	12.95234	94.07876	100.00000	0.00000	0.00000	0.00000	2.09153	1.00018
xrp	0.11132	6.29600	14.30666	67.40079	0.00000	0.00000	0.00000	0.00000	1.02617	1.04262
solana	0.14842	5.38926	15.06570	66.55776	0.00000	0.00000	0.00000	0.00000	1.29304	161.68237
...	...	...	...	...	...	...	...	...	...	...
dinero	99.62894	0.00000	14.58926	0.00000	0.00000	0.00000	0.00000	100.00000	1.79390	0.00034
mox	99.77737	0.00000	14.58926	0.00000	0.00000	0.00000	0.00000	100.00000	2.19489	0.00048
argus	99.88868	0.00000	8.67168	0.00000	0.00000	0.00000	0.00000	100.00000	0.48514	0.00144
ouroboros	99.96289	0.00001	6.81690	0.00000	0.00000	0.00000	0.00000	0.00000	0.88809	0.00005
kz-cash	100.00000	0.00000	15.41115	0.00000	0.00000	0.00000	0.00000	100.00000	0.99695	0.00091

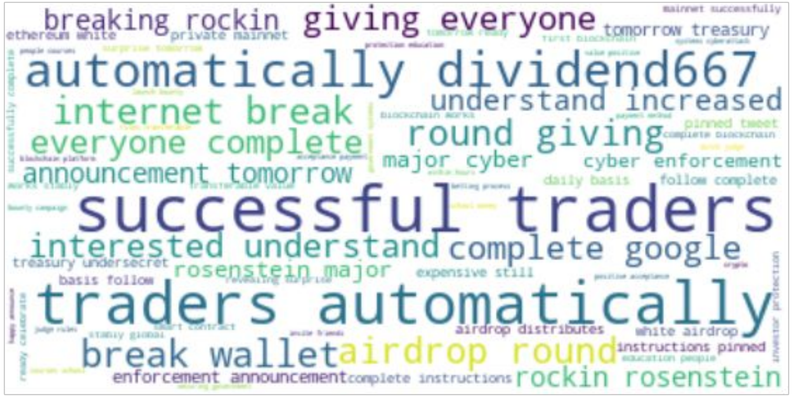
1628 rows × 10 columns

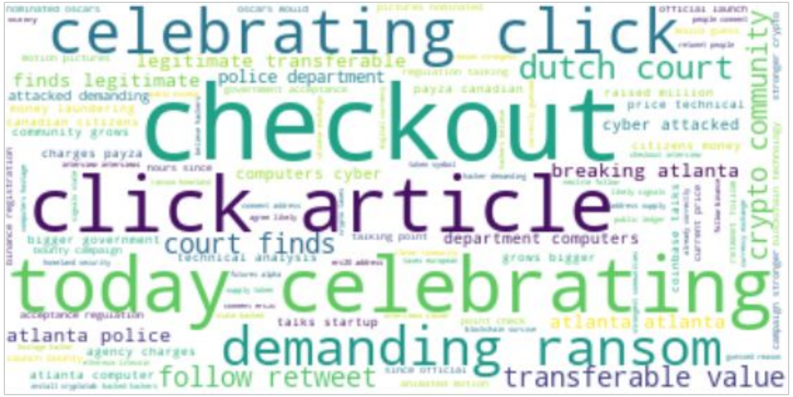
Analysis of Tagged Bitcoin-related Tweet Dataset

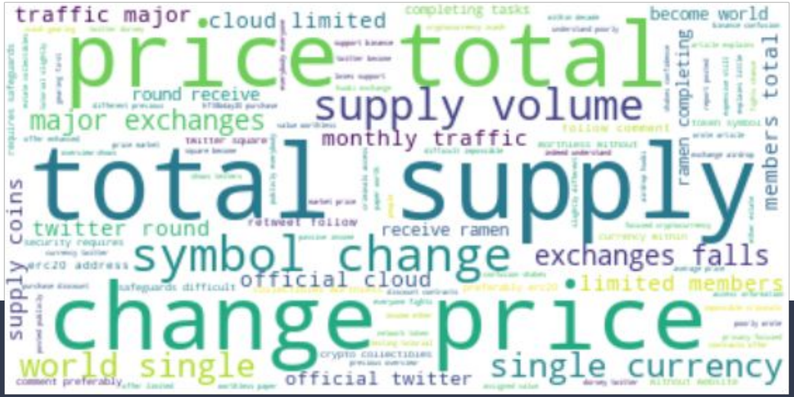
- Observe an unbalanced distribution among different sentiment categories (solved by oversampling later)



Visualization of Frequent Words


Word Cloud for Positive Tweets


Word Cloud for Neutral Tweets


Word Cloud for Negative Tweets

Methodology

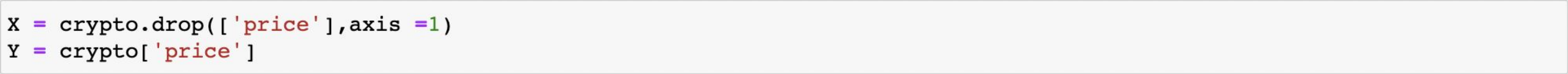
Prediction: Overview

Target variable: price
Input variable(9) : cmc_rank, volume_24h, percent_change_7d, market_cap, 'binance-smart-chain', 'collectibles-nfts','defi', 'mineable', circulating/total_supply.

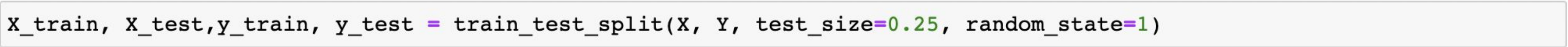
We have tried 4 models:

- **Linear Regression**
- **Polynomial Regression**
- **Lasso Regression**
- **Ridge Regression (The Best model)**

Train-Test split (test size 0.25)



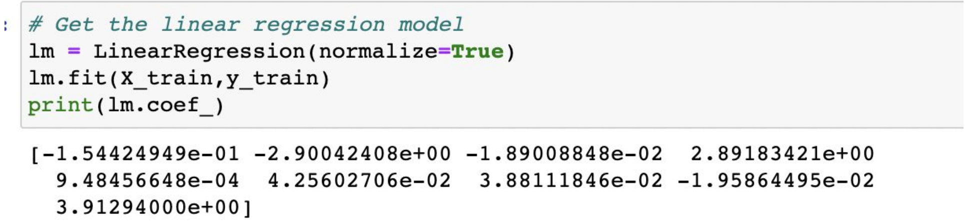
```
X = crypto.drop(['price'],axis =1)
Y = crypto['price']
```



```
X_train, X_test,y_train, y_test = train_test_split(X, Y, test_size=0.25, random_state=1)
```

Prediction

- **Linear Regression**



```
# Get the linear regression model
lm = LinearRegression(normalize=True)
lm.fit(X_train,y_train)
print(lm.coef_)
```

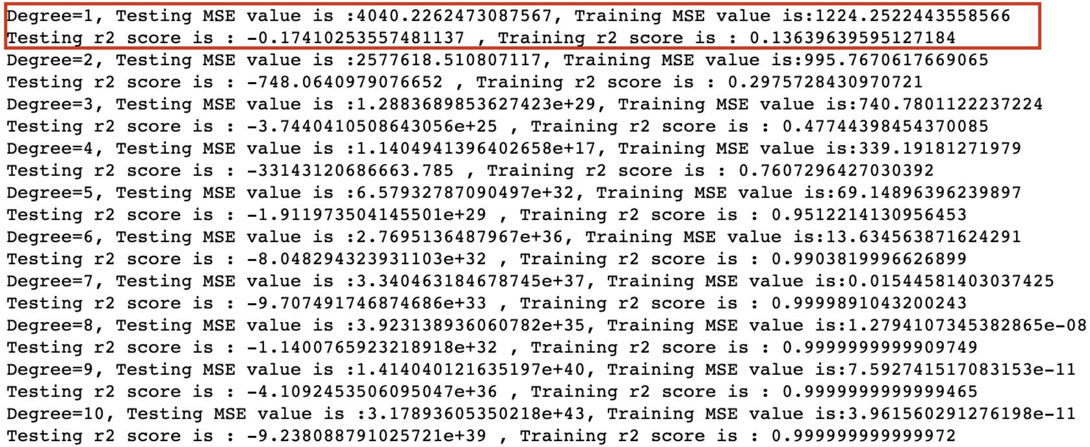
```
[-1.54424949e-01 -2.90042408e+00 -1.89008848e-02  2.89183421e+00
  9.48456648e-04  4.25602706e-02  3.88111846e-02 -1.95864495e-02
  3.91294000e+00]
```

- **Linear Regression Result:**
  Training R-squared: 0.13639639595127184
  Testing R-squared: -0.17410253557481048
  Training Mean squared error: 1224.25
  Testing Mean squared error: 4040.23

- **Training R-squared is ok (>0, because financial asset (especially cryptocurrencies) returns are often said to be unpredictable.).**
- **MSEs are both very large.**

Prediction

- **Polynomial**
  Use the training dataset to find the best degree
- **Result:**



```
Degree=1, Testing MSE value is :4040.2262473087567, Training MSE value is:1224.2522443558566
Testing r2 score is : -0.17410253557481137 , Training r2 score is : 0.13639639595127184
Degree=2, Testing MSE value is :2577618.510807117, Training MSE value is:995.7670617669065
Testing r2 score is : -748.0640979076652 , Training r2 score is : 0.2975728430970721
Degree=3, Testing MSE value is :1.2883689853627423e+29, Training MSE value is:740.7801122237224
Testing r2 score is : -3.7440410508643056e+25 , Training r2 score is : 0.47744398454370085
Degree=4, Testing MSE value is :1.1404941396402658e+17, Training MSE value is:339.19181271979
Testing r2 score is : -33143120686663.785 , Training r2 score is : 0.7607296427030392
Degree=5, Testing MSE value is :6.57932787090497e+32, Training MSE value is:69.14896396239897
Testing r2 score is : -1.911973504145501e+29 , Training r2 score is : 0.9512214130956453
Degree=6, Testing MSE value is :2.7695136487967e+36, Training MSE value is:13.63456387162429l
Testing r2 score is : -8.048294323931103e+32 , Training r2 score is : 0.9903819996626899
Degree=7, Testing MSE value is :3.340463184678745e+37, Training MSE value is:0.01544581403037425
Testing r2 score is : -9.707491746874686e+33 , Training r2 score is : 0.9999891043200243
Degree=8, Testing MSE value is :3.923138936060782e+35, Training MSE value is:1.2794107345382865e-08
Testing r2 score is : -1.1400765923218918e+32 , Training r2 score is : 0.9999999999909749
Degree=9, Testing MSE value is :1.414040121635197e+40, Training MSE value is:7.592741517083153e-11
Testing r2 score is : -4.1092453506095047e+36 , Training r2 score is : 0.9999999999999465
Degree=10, Testing MSE value is :3.17893605350218e+43, Training MSE value is:3.961560291276198e-11
Testing r2 score is : -9.238088791025721e+39 , Training r2 score is : 0.999999999999972
```

Degree = 1 it is most appropriate.

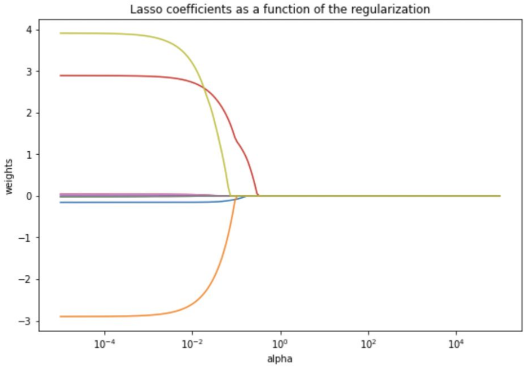
Testing R-squared value is better than that of other degrees.

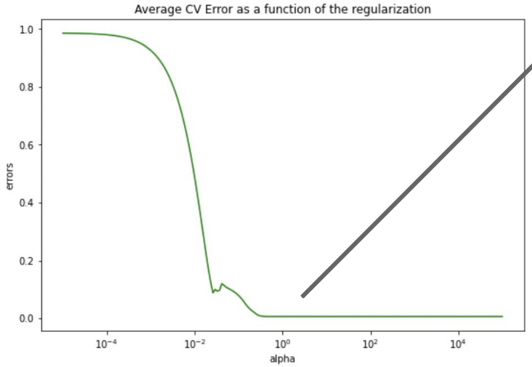
Training MSE is lower when degree = 2 than when degree = 1, but by balancing the training and testing result, degree = 1 is still our choice. When degree = 1, it means linear regression is better than polynomial features.

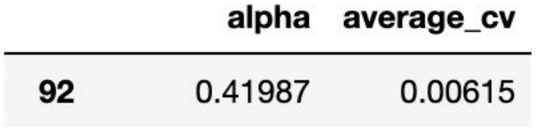
So they are equally not promising in doing the prediction.

Prediction

- Lasso:
  Use the training set to find the best alpha


Lasso coefficients as a function of the regularization


Average CV Error as a function of the regularization



	alpha	average_cv
92	0.41987	0.00615

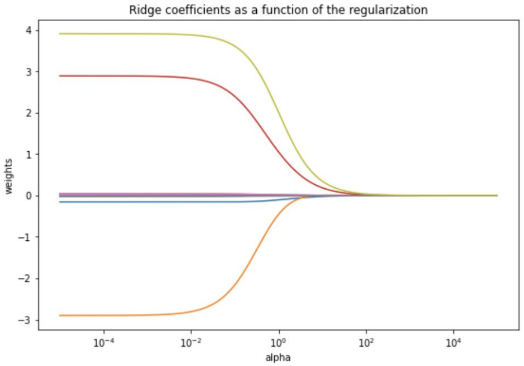
The alpha that corresponds to the smallest average_cv

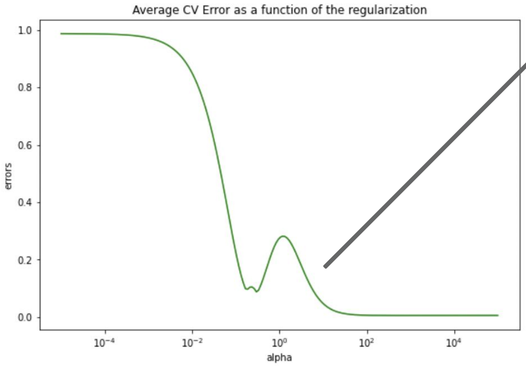
fit the test set

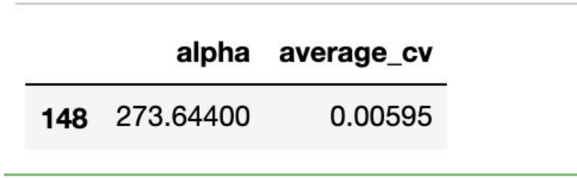
Testing MSE :3474.0727

Prediction

- Ridge:
  Use the training set to find the best alpha







	alpha	average_cv
148	273.64400	0.00595

The alpha that corresponds to the smallest average_cv

fit the test set

Testing MSE :3474.8058

Prediction: Conclusion

Moel	MSE
Linear Regression	4040.23
Polynomial Regression(degree = 1)	4040.23
Lasso Regression(alpha = 0.41987)	3474.805809882059
Ridge Regression(alpha =273.64400) Best Model	3474.0727258497113

We choose to use Ridge Regression (alpha =273.64400) model to do prediction on cryptocurrency prices. By predicting the price of different cryptocurrencies, we can compare our model's prediction with the current price of cryptocurrencies on the market and identify if any crypto is over-priced or under-priced.

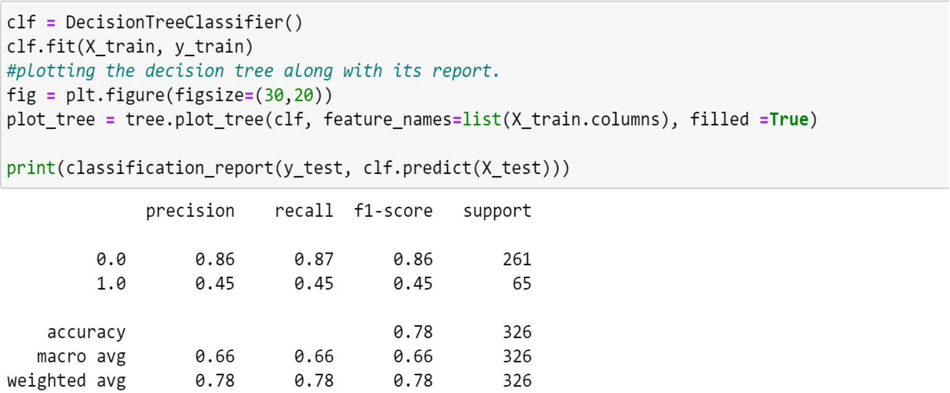
Classification- Overview

- Question :Inspecting whether cryptocurrencies with the same technical properties('mineable') have similarities in pricing, percent change, volume etc.

- By further exploring the data, we decided to create classification models to classify cryptocurrencies on the target dummy variables of 'mineable', specifically:
  - Decision Tree Classifier
  - Random Forest
  - KNN
- Within the cryptocurrency world, there is a vast array of technical properties that affect the market behavior of cryptocurrencies, and mineable is one of them.
- Mineable: indicates whether a cryptocurrency can be acquired through mining.
- The value of those cryptos, from a finance point of view, has a more stable value and growth, which can be reflected through percent price change, and consistent growth in volume.

Classification- Decision Tree Classifier

- Decision Tree Classifier with no limit on depth reached a accuracy of 0.78

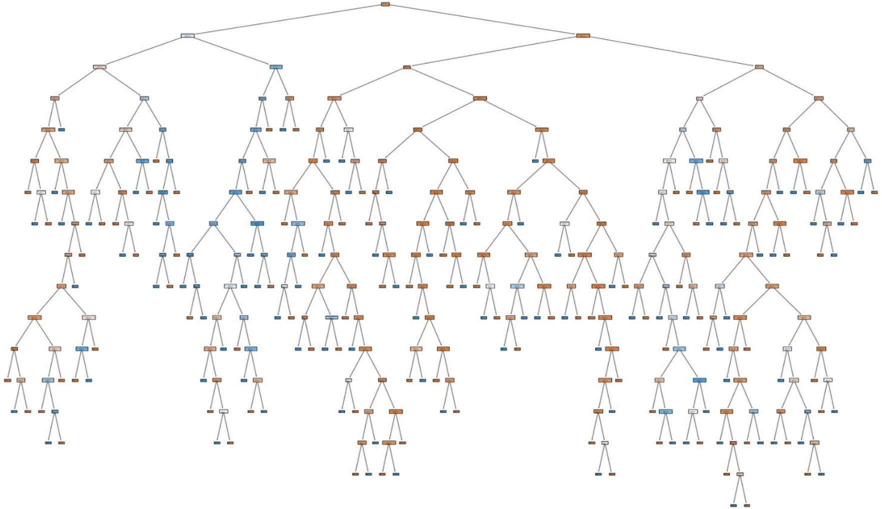


```
clf = DecisionTreeClassifier()
clf.fit(X_train, y_train)
#plotting the decision tree along with its report.
fig = plt.figure(figsize=(30,20))
plot_tree = tree.plot_tree(clf, feature_names=list(X_train.columns), filled =True)

print(classification_report(y_test, clf.predict(X_test)))
```

```
              precision    recall  f1-score   support

         0.0       0.86      0.87      0.86       261
         1.0       0.45      0.45      0.45        65

    accuracy                           0.78       326
   macro avg       0.66      0.66      0.66       326
weighted avg       0.78      0.78      0.78       326
```

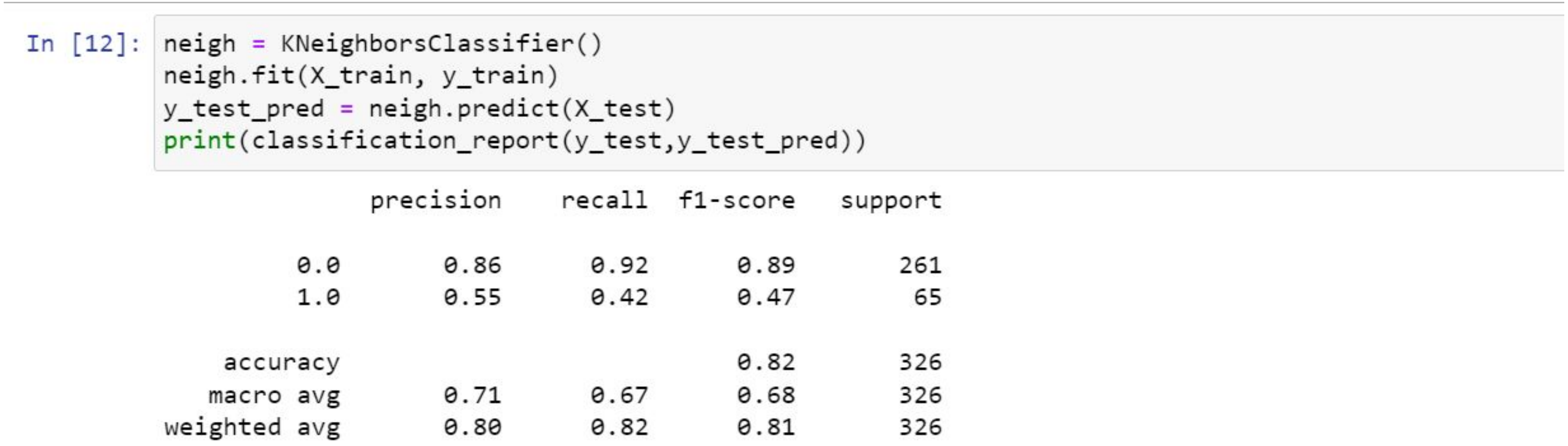
Classification– Decision Tree Classifier

- Since there is no limit on depth, the tree is fully extended.
- As we can see, the full tree is quite large.



Classification-KNN

- KNN reached a higher accuracy of 0.82, therefore we proceed to model tuning for KNN

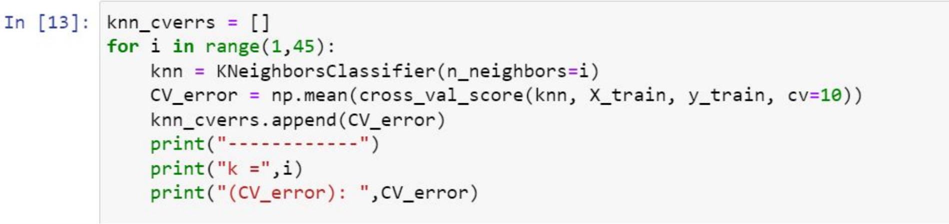


```
In [12]: neigh = KNeighborsClassifier()
         neigh.fit(X_train, y_train)
         y_test_pred = neigh.predict(X_test)
         print(classification_report(y_test,y_test_pred))
```

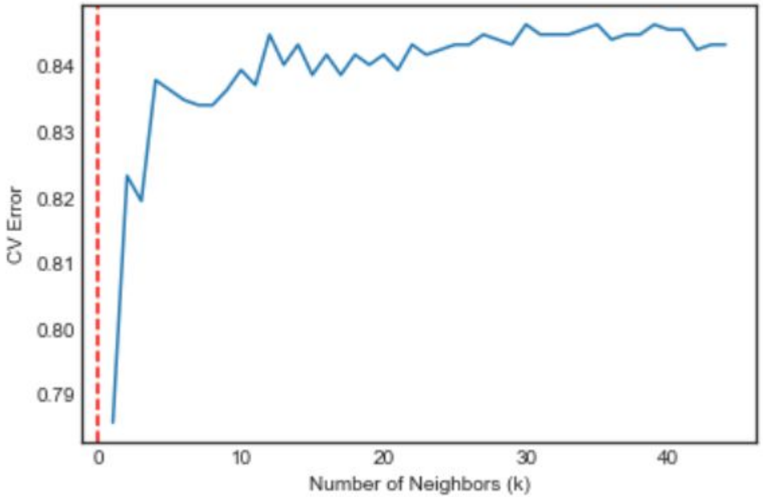
```
              precision    recall  f1-score   support

         0.0       0.86      0.92      0.89       261
         1.0       0.55      0.42      0.47        65

    accuracy                           0.82       326
   macro avg       0.71      0.67      0.68       326
weighted avg       0.80      0.82      0.81       326
```

Classification-KNN

- By tuning the number of neighbors, we see that the model has the lowest Cross Validation Score when k =1, yielding a accuracy of 0.81

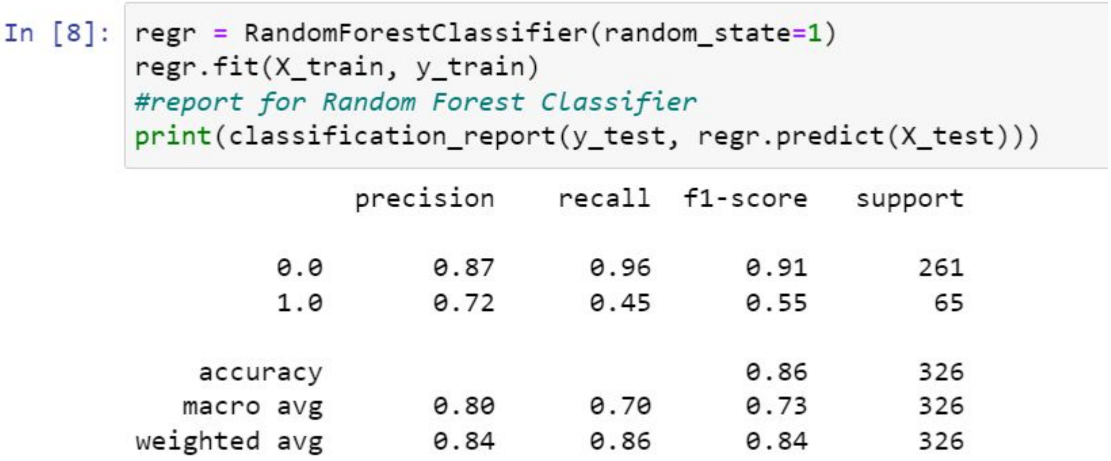


```
In [13]: knn_cverrs = []
         for i in range(1,45):
             knn = KNeighborsClassifier(n_neighbors=i)
             CV_error = np.mean(cross_val_score(knn, X_train, y_train, cv=10))
             knn_cverrs.append(CV_error)
             print("------------")
             print("k =",i)
             print("(CV_error): ",CV_error)
```



Classification-Random Forest

- Random Forest reached so far the best accuracy of 0.86.

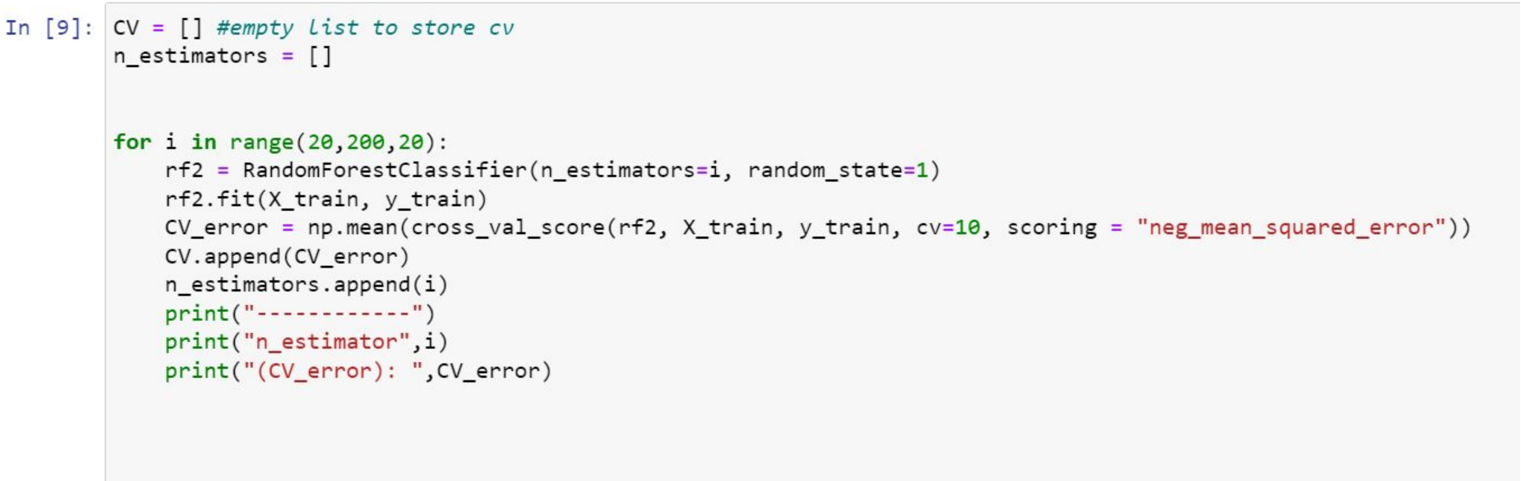


```
In [8]: regr = RandomForestClassifier(random_state=1)
        regr.fit(X_train, y_train)
        #report for Random Forest Classifier
        print(classification_report(y_test, regr.predict(X_test)))
```

```
              precision    recall  f1-score   support

         0.0       0.87      0.96      0.91       261
         1.0       0.72      0.45      0.55        65

    accuracy                           0.86       326
   macro avg       0.80      0.70      0.73       326
weighted avg       0.84      0.86      0.84       326
```

Classification-Random Forest

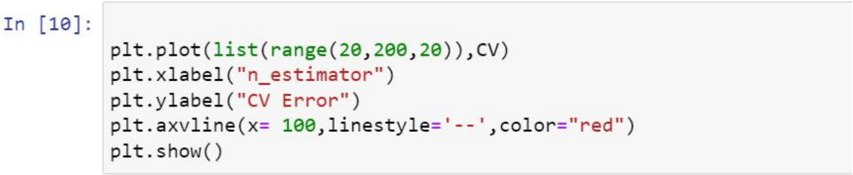
- We then tune the classification model by its number of estimators.



```
In [9]: CV = [] #empty list to store cv
        n_estimators = []


        for i in range(20,200,20):
            rf2 = RandomForestClassifier(n_estimators=i, random_state=1)
            rf2.fit(X_train, y_train)
            CV_error = np.mean(cross_val_score(rf2, X_train, y_train, cv=10, scoring = "neg_mean_squared_error"))
            CV.append(CV_error)
            n_estimators.append(i)
            print("------------")
            print("n_estimator",i)
            print("(CV_error): ",CV_error)
```

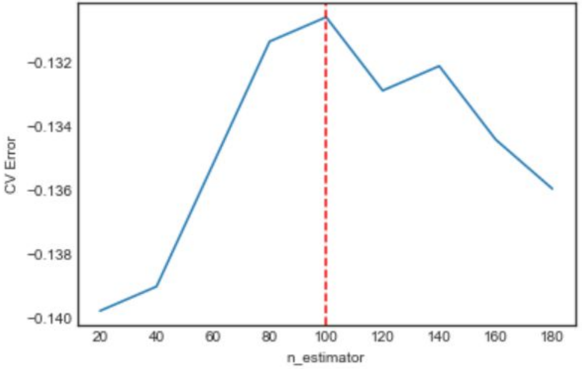
Classification-Random Forest

- Our best model with the lowest CV error is at n_estimators =100



In [10]:

```
plt.plot(list(range(20,200,20)),CV)
plt.xlabel("n_estimator")
plt.ylabel("CV Error")
plt.axvline(x= 100,linestyle='--',color="red")
plt.show()
```



Classification-Best Model

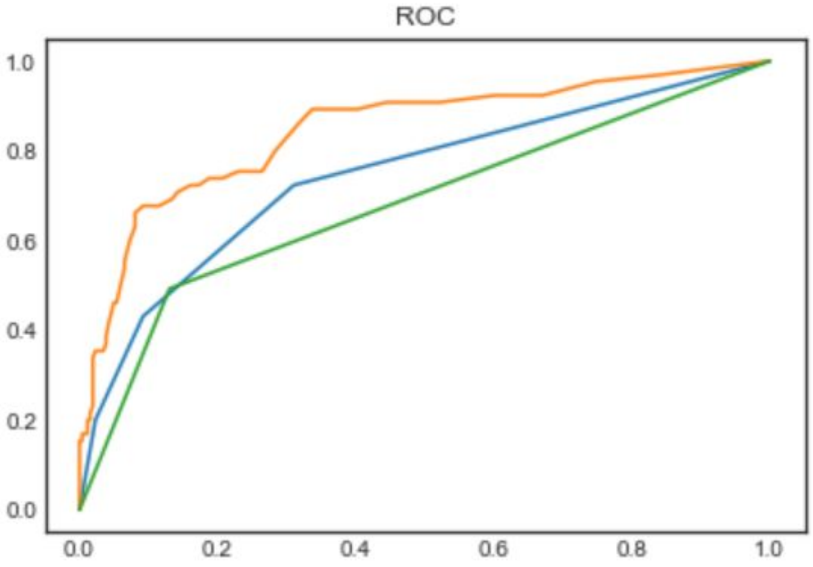
- Based on the accuracy of the best model of each category:

Model	**Accuracy**
Decision Tree Classifier	0.78
KNN	0.81
Random Forest	**0.86**

- Best Model is random Forest with 100 as its number of estimators

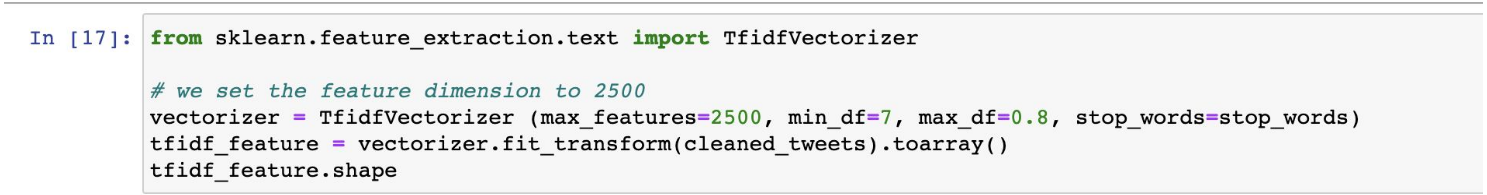
Classification–Best Model

- To confirm our best model, a ROC curve for each model is made:
  - Orange: Random Forest
  - Blue: KNN
  - Green: Decision Tree



Sentiment Analysis – Feature Extraction

- **TF-IDF (term frequency-inverse document frequency)** is a statistical measure that evaluates how relevant a word is to a document in a collection of documents.
- Establish a feature matrix of 2500 dimensions with TF-IDF Vectorizer from sklearn module



```
In [17]: from sklearn.feature_extraction.text import TfidfVectorizer

         # we set the feature dimension to 2500
         vectorizer = TfidfVectorizer (max_features=2500, min_df=7, max_df=0.8, stop_words=stop_words)
         tfidf_feature = vectorizer.fit_transform(cleaned_tweets).toarray()
         tfidf_feature.shape
```

- Oversampling the dataset: due to the imbalanced feature of the current dataset, we would like to generate new samples in the classes which are under-represented with the help of the imblearn module.

Sentiment Analysis – Model Selection and Tuning

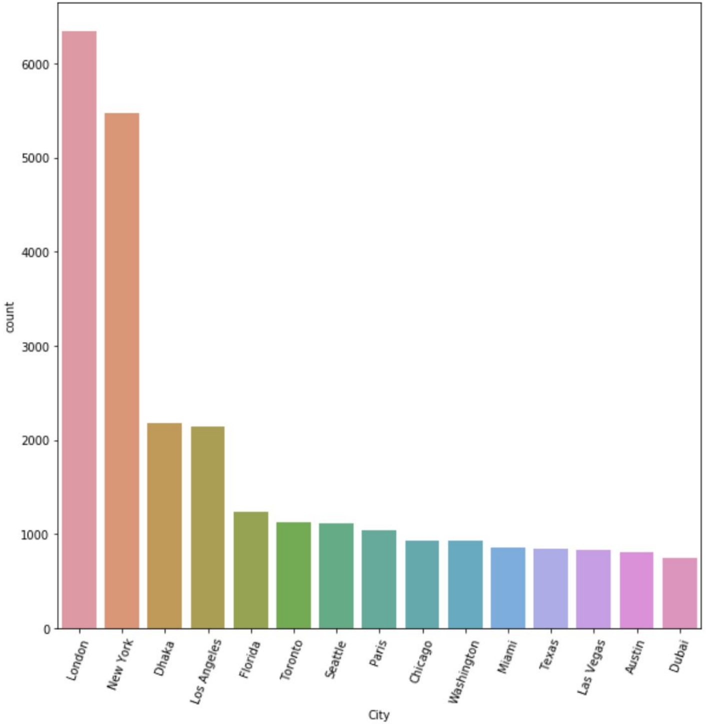
1. We tried with four different classification models:
    a. Logistic Regression 94% accuracy -- good performance and low complexity
    b. Naive Bayes 79% accuracy -- unsatisfactory accuracy score
    c. Decision Tree 96% accuracy -- runtime is too long, good performance
    d. Random Forest 97% accuracy -- runtime is too long, best performance

2. We tuned Logistic Regression with 3 different solvers: ['newton-cg', 'lbfgs', 'liblinear']
    a. Liblinear results in the lowest cv error

3. We tuned Random Forest on 3 major hyperparameters: n_estimators, max_depth, min_samples_split

Sentiment Analysis - Classification on Unseen Dataset

- Import another unlabelled dataset for real-world classification
- Utilize two models (tuned Logistic Regression and tuned RF) to generate two new columns of classification results
- Among 299828 rows, there are 79182 rows have different classification results between RF and Logistic Regression. The unmatching rate is 26%.
- Extract valid user location information to analyze how sentiment varies across different cities



Valid Tweets Count in Major Cities

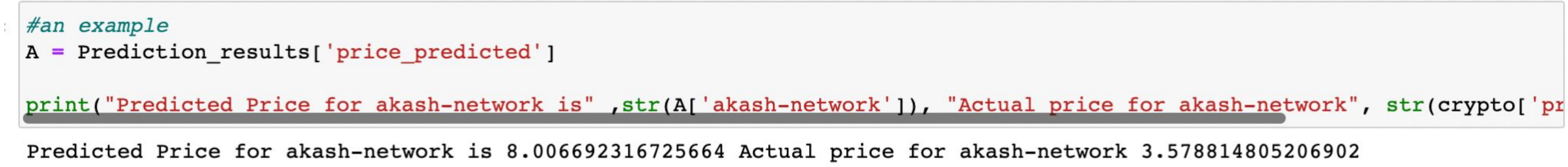
Conclusions

Prediction-Conclusion

- **By comparing the MSEs, we see that the best model for prediction is Ridge Regression Model.**

- With this model , we can compare our model's prediction with the current price of cryptocurrencies on the market and identify if any crypto is overpriced or underpriced. We can invest into the cryptos that are undervalued, and short the ones that are over-valued, and anticipate potential market correction.

Example :



```
#an example
A = Prediction_results['price_predicted']

print("Predicted Price for akash-network is" ,str(A['akash-network']), "Actual price for akash-network", str(crypto['pr
```

```
Predicted Price for akash-network is 8.006692316725664 Actual price for akash-network 3.578814805206902
```

- **We should buy more akash-network.**

Classification-Conclusion

- By comparing the accuracy of our models and looking at the ROC graph, we see that random forest has the best performance of 0.86 accuracy. Therefore, this is the most appropriate model to answer our question.

- With this accuracy, we can conclude that for a cryptocurrency to be mineable, it does show behavior similarities within our variable space such as volume, pricing change etc.

- By using our classification model, we can test whether mineable cryptos show similar behaviors with others that has the mineable tag. If a crypto is mineable and is classified as not, we can question the integrity of the blockchain foundation, and see why it is not showing similar market behavior

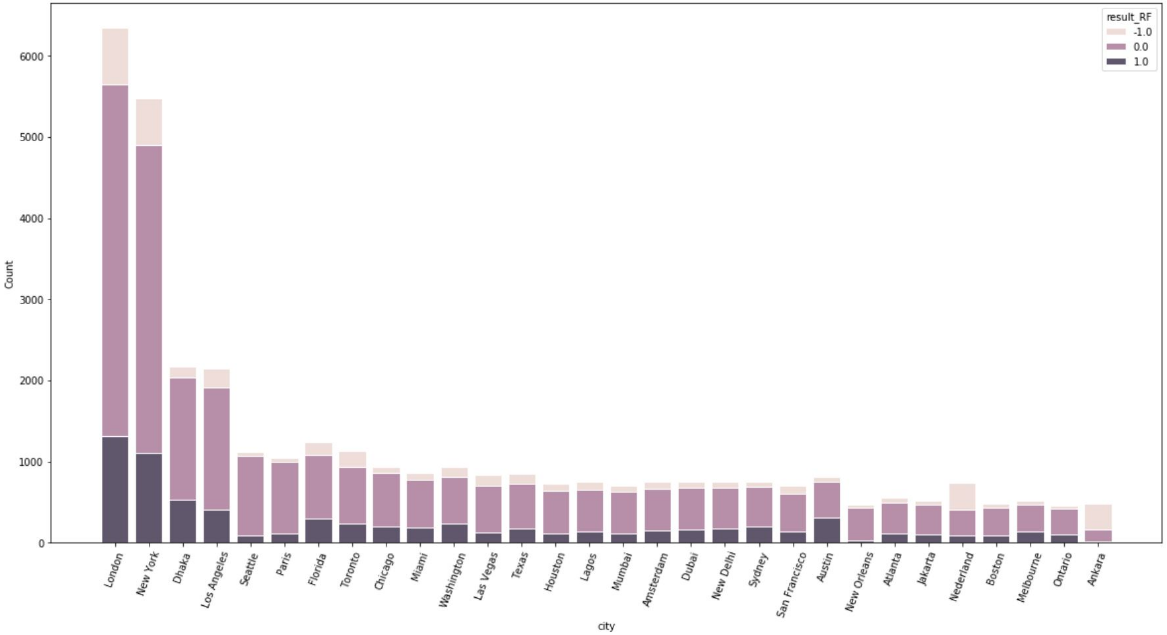
Sentiment Analysis(Classification) -Conclusion

1. By categorizing tweets into positive, neutral, and negative categories and calculating the negative, positive, and opinionative rate of each city, we are able to identify cyber citizens from which regions care most about Bitcoin market and are more likely to express negative/positive sentiments.

2. Random Forest shows the best performance, while Decision Tree and Logistic Regression performs considerably well. However, it takes a really long time for RF to run.

Sentiment Analysis(Classification) –Conclusion

1. The figure shows the sentiment distribution of top 30 active cities concerning Bitcoin tweets.
2. People in London and New York highly active on Twitter about Bitcoin.
3. Chicago, Washington, Austin, Florida, and Dhaka are the top 5 opinionative cities towards Bitcoin.
4. Posts provide valuable sentiment information to predict the price fluctuations of cryptocurrencies.



Future Work

FUTURE WORK

Sentiment Analysis:

- We only focus on TF-IDF to select features, however, it is worth investigating how to use **other feature selection techniques, such as Mutual information, Chi-Square, and Information gain to select features from high dimensionality of feature set.**
- Also, it will be useful to consider **unigram, bigram, POS tags of words, function words, and particularity of microblogging texts in our feature set.**
- We should conduct **more thorough hyperparameter tuning for Random Forest Classifier.**
- Analyze the whole unlabelled dataset, instead of selecting subset, to gain a broader understanding

Prediction & Classification:

- **Data constraints:** Since our data is only focused on cryptos with prices less than 500, because larger cryptocurrencies show very different behaviors, **in the future we can potential make different models for different price ranges.**

Prediction:

- It is also worth exploring using **decision tree to predict the price**.
- Another way to explore the prices of cryptocurrencies is through **Time Series Analysis**. It is possible for us to get the time series data of the cryptocurrencies and use time series algorithms such as ARIMA to predict its prices. It is also possible to implement the unsupervised learning on the time series data to generate more insights.
- There could also be other variables that can play a major role in our prediction, we could attempt to find additional data through web scraping, or exploring forums.

Classification:

- We can find more data within each category of crypto, we could perhaps **make individual models and assumptions based on different tags such as NFT, DAO, defi** etc.

REFERENCE

Brownlee, J. (2020, August 27). *Tune hyperparameters for Classification Machine Learning Algorithms*. Machine Learning Mastery. Retrieved December 9, 2021, from https://machinelearningmastery.com/hyperparameters-for-classification-machine-learning-algorithms/.

Over-sampling - Version 0.8.1. (n.d.). Retrieved December 9, 2021, from https://imbalanced-learn.org/stable/over_sampling.html.