Project Pandemic:

Backend :

Market Data API Host
Us-east-1a
ec2-34-207-72-173.compute-1.amazonaws.com
34.207.72.173

Epidemic Data API Host
us-east-1a
ec2-3-85-238-79.compute-1.amazonaws.com
3.85.238.79

Market Data DB Host
us-east-1b
database-1.cpnwuinodiya.us-east-1.rds.amazonaws.com

Epidemic Data DB Host
us-east-1
corona.cuo7ivhfh3jn.us-east-1.rds.amazonaws.com

API endpoints:
http://34.207.72.173/market - returns last 100 days of market data for Dow Jones, S&P, FTSE100, Nikkei and Hang Seng Index in json format
http://34.207.72.173/getCSV - returns all available market data in csv format

http://3.85.238.79//api/v1.0/pandemic/corona - returns coronavirus case informations including confirmed cases and deaths by date and location in json format
http://3.85.238.79//api/v1.0/pandemic/getCSV - returns all available epidemic data in csv format

COVID Dataset ETL:

1. Ingest CSV file from (https://console.aws.amazon.com/dataexchange/home?region=us-east-1#/products/prodview-gsdi4ujyb4gfy) into python
2. Dropped any unnecessary columns and renamed columns for easier use.
3. Pivoted the Case Type and Cumulative Cases columns so we can have them as rows for easier used in upcoming analysis
4. Dropped any rows that contained zero's across all metric columns in order to get rid of unnecessary rows and shrink file size.
5. Saved both 2 clean copies (1) original file with dropped and renamed columns (2) pivoted data.

Twitter Sentiment Analysis:

Collected the Corona tweets for March 30th. We have applied for API access but never got approved by Twitter.

Data Cleaning: (Step 1)
(https://github.com/maheshdivan/Project_Pandemic/blob/master/Tweet_Sent_Analysis/tweet_sentiment_analysis.ipynb)

Loaded data into S3 for storage
Used Pyspark to perform data cleaning activity on the tweets.

Steps:

1. Removed unnecessary columns
2. Using regex removed URL links User IDs and Hash tags from tweets
3. Removed non-English words from tweet

Sentiment Analysis (Step 2)
(https://github.com/maheshdivan/Project_Pandemic/blob/master/Tweet_Sent_Analysis/run_sent.py)

Libraries used are

Python Natural Language Toolkit (nltk)
SklearnClassifier
Pickle
Sklearn.Naive Bayes
Sklearn linear model
Sklearn SVM
Nltk tokenize

Method used for Sentiment Analysis: (Arriving at a model)

Created a text file of positive and negative reviews (These are actually taken from movie reviews)
Using above file, using nltk parts of speech module (nltk.pos_tag), tagged each word, which will give each word a tag with parts of speech

```
POS tag list:
CC   coordinating conjunction
CD   cardinal digit
DT   determiner
EX   existential there (like: "there is" ... think of it like "there exists")
FW   foreign word
IN   preposition/subordinating conjunction
JJ   adjective  'big'
JJR  adjective, comparative     'bigger'
JJS  adjective, superlative     'biggest'
LS   list marker     1)
MD   modal could, will
NN   noun, singular 'desk'
NNS  noun plural     'desks'
NNP  proper noun, singular 'Harrison'
NNPS proper noun, plural   'Americans'
PDT  predeterminer   'all the kids'
POS  possessive ending    parent\'s
PRP  personal pronoun     I, he, she
PRP$ possessive pronoun   my, his, hers
RB   adverb     very, silently,
RBR  adverb, comparative  better
RBS  adverb, superlative  best
RP   particle   give up
TO   to    go 'to' the store.
UH   interjection    errrrrrrrm
VB   verb, base form take
VBD  verb, past tense     took
VBG  verb, gerund/present participle taking
VBN  verb, past participle taken
VBP  verb, sing. present, non-3d     take
VBZ  verb, 3rd person sing. present  takes
WDT  wh-determiner   which
WP   wh-pronoun who, what
```

```
WP$   possessive wh-pronoun whose
WRB   wh-abverb  where, when
```

From the above list, we were only interested in words which are Adjectives, which are part of speech words starting with "J"

Performed above step for positive and negative reviews and stored this as document.pickle files. (Pickle is used to store these for faster training)

Created a frequency distribution of words for selecting 5000 words, store this as well as pickle file

Created a feature set using documents from above list and create a dictionary.

From the featureset create a testing and training set and use above for training each of the algorithms used

1. nltk.NaiveBayesClassifier
2. SklearnClassifier(MultinomialNB()
3. SklearnClassifier(BernoulliNB()
4. SklearnClassifier(LogisticRegression()
5. SklearnClassifier(LinearSVC()
6. SklearnClassifier(SGDClassifier()

Pickle the models from all the above step for predicting a new dataset

Above steps were used for creating the initial model

Sentiment Analysis Module:

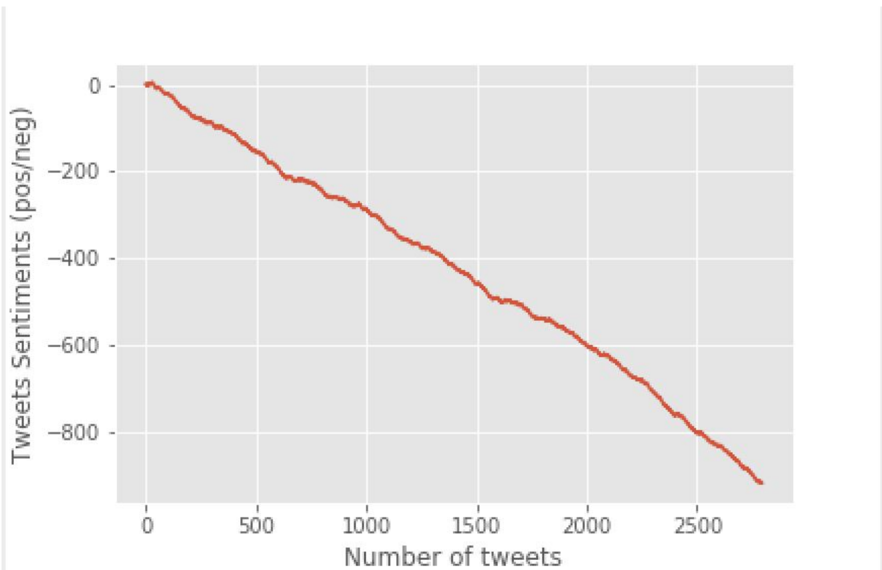
https://github.com/maheshdivan/Project_Pandemic/blob/master/Tweet_Sent_Analysis/sentiment_mod.py

Created a module sentiment_mod.py using above steps and added a voting for each classified which will give a confidence value for each classifier. In this module all the detailed steps were removed by using picked modules from initial steps mentioned above.

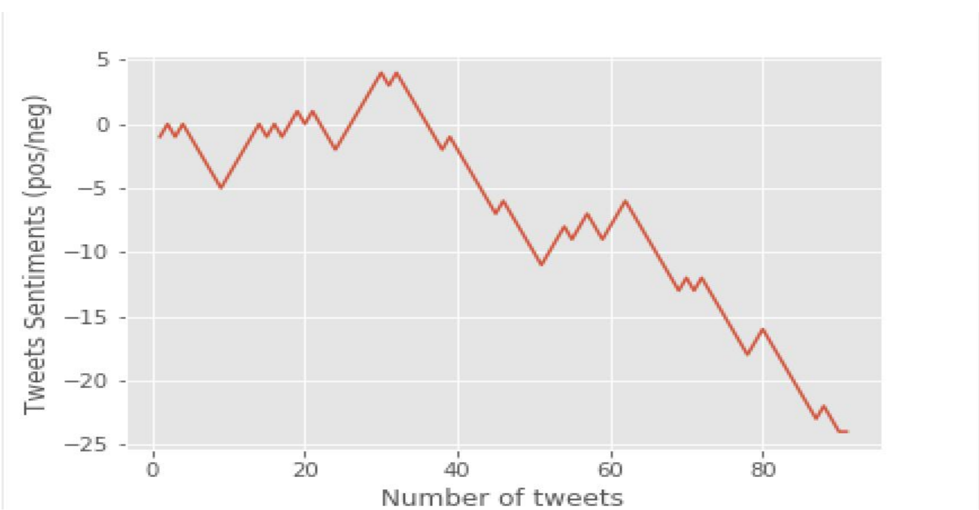
Sentiment Analysis and Plotting: (Using model for predicting and Plotting): (Step 2)

Using the tweets CSV file, used 2500 tweets to derive the sentiment out of each tweet from the cleaned data set from step 1. Selected the tweets which has more than 80% confidence level for writing into a tweet sentiment analysis file for plotting. Using this file, plot the sentiment by assigning +1 for positive sentiment and -1 for each negative sentiment.

Twitter Sentiment Analysis :



Ran for 100 tweets to see the variation in positive and negative sentiment



Corona ML:

(https://github.com/maheshdivan/Project_Pandemic/tree/master/Corona_Curve_ML)

We have used same dataset used for visualization which has been cleaned using python data cleaning
(https://github.com/maheshdivan/Project_Pandemic/blob/master/ETL/ETL-covid19.ipynb)

For predicting the when curve will flatten, we have used sigmoid function.

Approach:

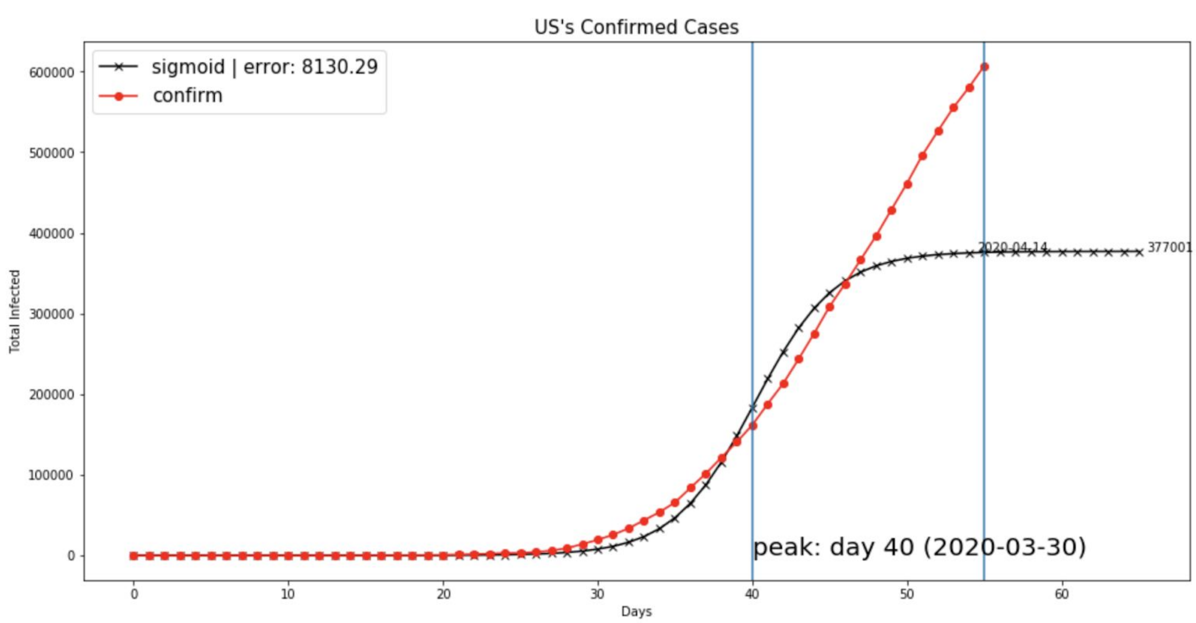
From the cleaned dataframe, removed the unnecessary columns like confirmed and deaths as we will be using cumulative data for prediction.

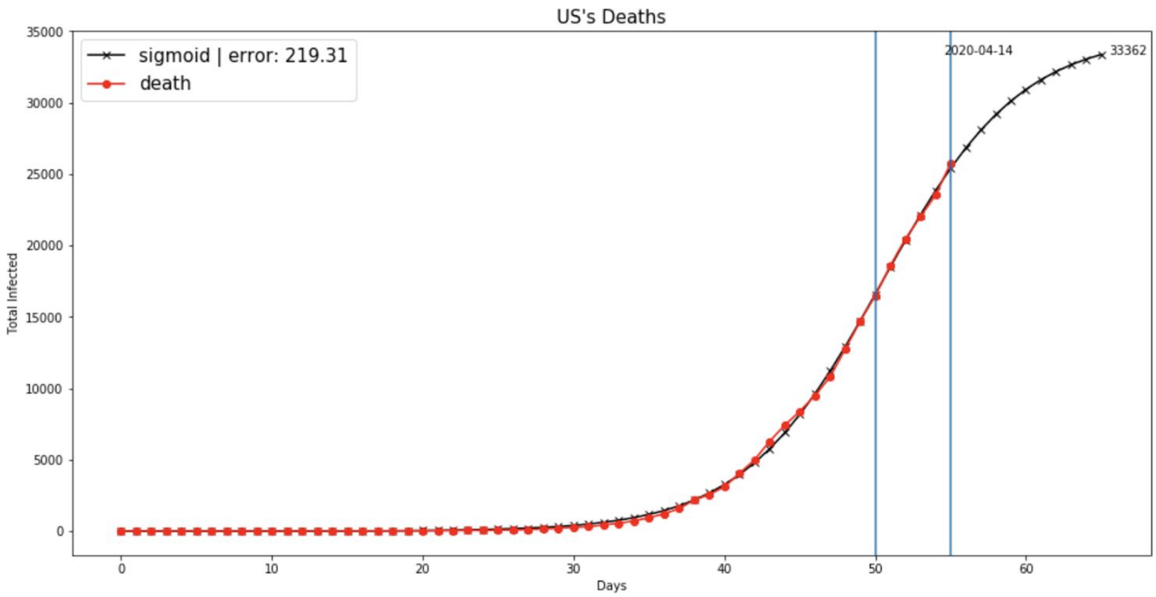
Created a sigmoid function and using scipy curve fit function, calculated popt and pcov. We used popt value to calculate the peak day for confirmed cases and deaths. We have used the 'dogbox' algorithm for minimization.

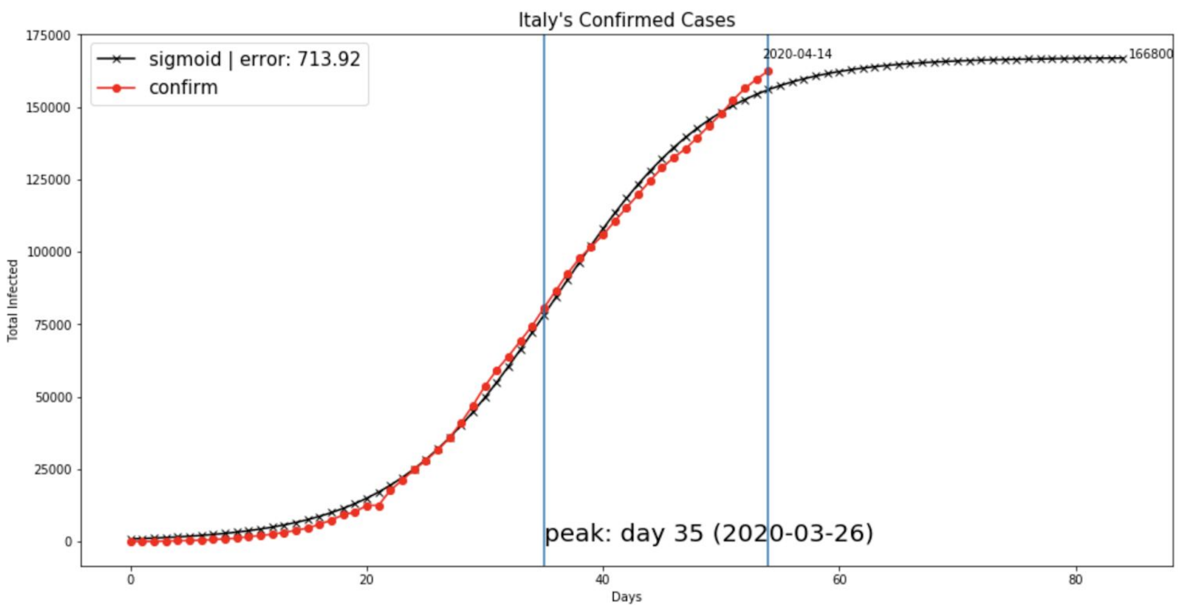
Variable inception is used based on data when most of the cases started to arrive. For "US", we have set it as 28 days from the start of the dataset. For Italy, it is set as 20. For China it is zero.

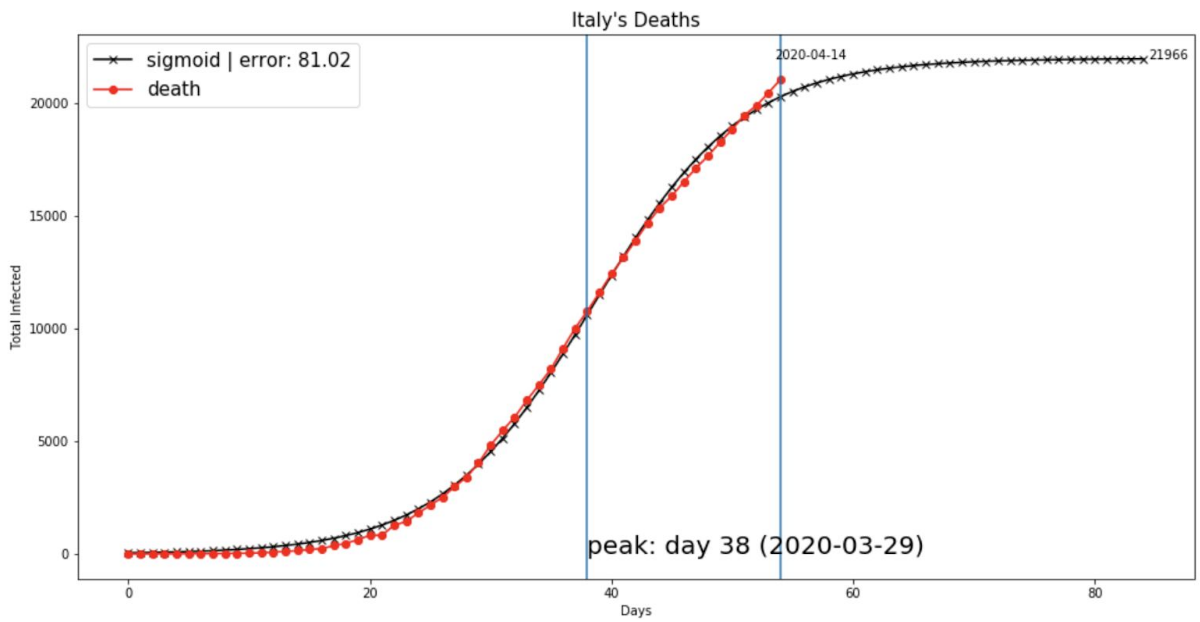
Curve fit scipy function needs bounds for the sigmoid curve this has been taken by trial and error method because too many big values give error as the curve ft function has reached maximum value.

Then plotted the actual and the Sigmoid curve to see how actual and predicted curves look.




US's Deaths
sigmoid | error: 219.31
death
2020-04-14
33362
Total Infected
Days


Italy's Confirmed Cases
sigmoid | error: 713.92
confirm
2020-04-14
166800
peak: day 35 (2020-03-26)
Total Infected
Days



Predictions of confirmed cases and death are done based on sigmoid function

Confirmed cases (US):

```
Predictions:
2020-04-15: 376134
2020-04-16: 376417
2020-04-17: 376611
2020-04-18: 376744
2020-04-19: 376835
2020-04-20: 376897
2020-04-21: 376939
2020-04-22: 376968
2020-04-23: 376988
```

Deaths (US):

```
Predictions:
2020-04-15: 26809
2020-04-16: 28066
2020-04-17: 29165
2020-04-18: 30113
2020-04-19: 30921
2020-04-20: 31603
2020-04-21: 32174
2020-04-22: 32649
2020-04-23: 33040
```