To get the tweets from a particular date up to some date. I will use getOldTweets3 instead of tweepy as tweepy has some restriction for the amount of tweets you can collect at a time.

```
pip install getOldTweets3    #to install the library
```

In GetOldTweets3 we can apply filter as #lockdown, #coronavirus, #covid19 to search for tweets only for that type. We can also add from and since date. In our project we are collecting data from 1 March,2020 to 31 May,2020.

```
got.manager.TweetCriteria().setQuerySearch('#covid19#coronavirus#lockdown').setSince("2020-03-01").setUntil("2020-05-31").setMaxTweets(700) # setMaxTweets is to set max tweets we are about to fetch
```

Now our next work is to get data which is inside India and also to separate data based on Indian states.
For this we have used another property of GetOldTweets3 which search on the basis of coordinates and within a particular radius of that coordinates.
We can easily get the location coordinates of each state but for radius we first find the area of that particular state.
https://en.wikipedia.org/wiki/List_of_states_and_union_territories_of_India_by_area

To find the radius we used the formula of □$r^2$

Now we stored all things into array named radius and coordinates for each state.
After that we again fetched the tweets

```
for i in range(0,len(radius)):
    tweetCriteria = got.manager.TweetCriteria().setQuerySearch('#covid19#coronavirus#lockdown').setSince("2020-03-01").setUntil("2020-05-31").setNear(coordinates[i]).setWithin(radius[i]+"km").setMaxTweets(700)
```

To categorize on the basis of gender we will use the API which uses the name of author to find gender. GetOldTweets3 gives username instead of real name of user so for getting the real name we used tweepy api to get name from username

```
gender=requests.get('http://api.namsor.com/onomastics/api/json/gender/'+name[0]+'/'+name[1]).json()['gender']
```

Further to find the daily increase in corona cases we have used the data already collected by https://www.covid19india.org/ in order to find the increase in corona cases in state in which tweet was posted on that particular day.

Next we have apply some preprocessing to the collected tweets which includes the tweet text language translation to english and tweet's author classification whether they are personal opinions or the tweets by an organization. As we have to find the sentiments of the people so we have to remove the organizational tweets.

For translation we have used google translator python API

```
pip install googletrans # to install library
from googletrans import Translator
translator = Translator()
translated_text=translator.translate(tweet.text) #usage
```

For classifying the personal opinions and organizational tweets we used the standford NERT tagger based on the author's nameof tweet

```
pip install StanfordNERTagger
from nltk.tag import StanfordNERTagger
st = StanfordNERTagger('english.all.3class.distsim.crf.ser.gz',
                       'stanford-ner.jar',
                       encoding='utf-8')
classified_text = st.tag(tokenized_name) # This NERTagger(Name entity resolution tagger) uses tokenized word for the classification for which we have used nltk word_tokenize to convert word into tokens and identifies input name belongs to PERSON or not
```

**TEXT PREPROCESSING**

Now for all the preprocessing related to the texts like we had to install nltk libraries and also scikit learn for text feature extraction.

For lemmatization we use the nltk library for first converting the given sentences to token and then providing each token with it's wordNet tag. Then after giving the wordnet Tag we lemmatize each word according to the tag and join it to form the new lemmatized sentence.

Associating each word to the wordNet tag:

```python
import nltk
from nltk.stem import WordNetLemmatizer
from nltk.corpus import wordnet

lemmatizer = WordNetLemmatizer()

def wordnet_tag(nltk_tag):
    if nltk_tag.startswith('J'):
        return wordnet.ADJ
    elif nltk_tag.startswith('V'):
        return wordnet.VERB
    elif nltk_tag.startswith('N'):
        return wordnet.NOUN
    elif nltk_tag.startswith('R'):
        return wordnet.ADV
    else:
        return None
```

For lemmating each sentence in the text data:

```
nltk_tag_assign =
nltk.pos_tag(nltk.word_tokenize(sentence))

 wordnet_tag = map(lambda x: (x[0], wordnet_tag(x[1])),
nltk_tag_assign)
    lemmatized_sentence = []
    for word, tag in wordnet_tag:
        if tag is None:
            #ifthereareno tagappendthewordasitis
            lemmatized_sentence.append(word)
        else:
            #else use the tag to lemmatize the token

lemmatized_sentence.append(lemmatizer.lemmatize(word, tag))
    return " ".join(lemmatized_sentence)

```

We use the TfidVectorizer from sklearn.feature_extraction.text. The TFid Vectorizer also makes sure of the tokenization by using the min_df paramter where min_df represents the minimum frequency a word must have in order to be in token list.  For removing stop words we are passing stop words - coronavirus, covid19, lockdown, and,the,at,is,u=you,your we are removing them from tokens.

```
import sklearn
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer =
TfidfVectorizer(min_df=3,stop_words={'coronavirus','covid19'
,'lockdown','and','the','corona','at','is','you','your','#'
})
train_text = vectorizer.fit_transform(X_text)
```

## DIMENSIONALITY REDUCTION

We are using Truncated SVD to remove the features from our dataset by also keeping the variance of our dataset upto 95%. As our dataset contained some sparse values therefore we used truncated SVD in order to keep the major dimensions. By experimenting with values we got that out of 1156 features present initially only 200 dimensions contained 95.54% variance of the data. Therefore we transform our text dataset to only 200 dimensions.

```
from sklearn.decomposition import TruncatedSVD
tr = TruncatedSVD(200)
p = tr.fit_transform(train_text)
print(sum(tr.explained_variance_ratio_)
0.9554045183756755
```

Now as after dimensionality reduction the axis have very small values so we use minMax Scaler to scale it in the range of 0 to 1.

```
from sklearn import preprocessing
min_max_scaler = preprocessing.MinMaxScaler()
p = min_max_scaler.fit_transform(p)
```

## REMOVING OUTLIERS

Now we need to remove the outliers in the tranasformed axis, So we inter quantile range of 0.05 to 0.95. Anything above or below this range will be removed.

```
def remove_outliers(df):
    for col in df.columns:
        if (((df[col].dtype)=='float64') | ((df[col].dtype)=='int64')):
            percentiles = df[col].quantile([0.05,0.95]).values
            df[col][df[col] <= percentiles[0]] = percentiles[0]
            df[col][df[col] >= percentiles[1]] = percentiles[1]
        else:
            df[col]=df[col]
    return df

final_df=remove_outliers(df1)
```

After running the function we notice that there were no outliers according to the given range.

Now after we balance the dataset according to the number of states as each state should contain equal number of tweets so as to include the perspective of entire nation equally.

So for this we use OverSampling. We use RandomOverSampler for this purpose.

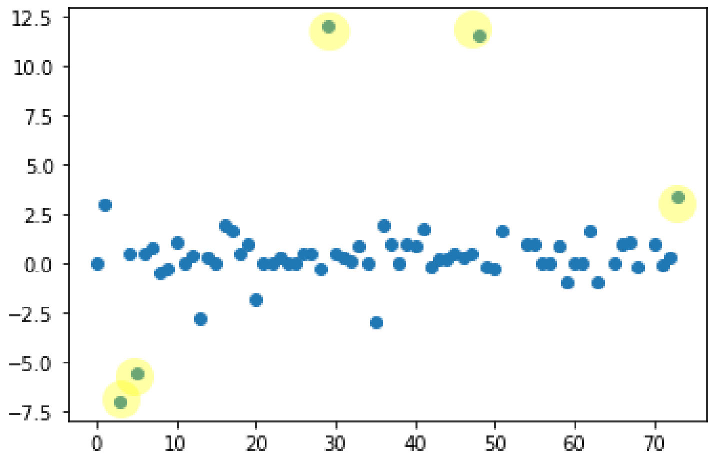
```
from imblearn.over_sampling import RandomOverSampler
sm1 = RandomOverSampler()
t,t1 = sm1.fit_resample(t,t1)
```

**TRAINING AND TESTING**

Now after all this we get a sentiments(balanced). Now we find the mean sentiment for each day, with weightage system that the text with more retweets is assigned more weightage. The formula to find mean sentiment for a day is as follows

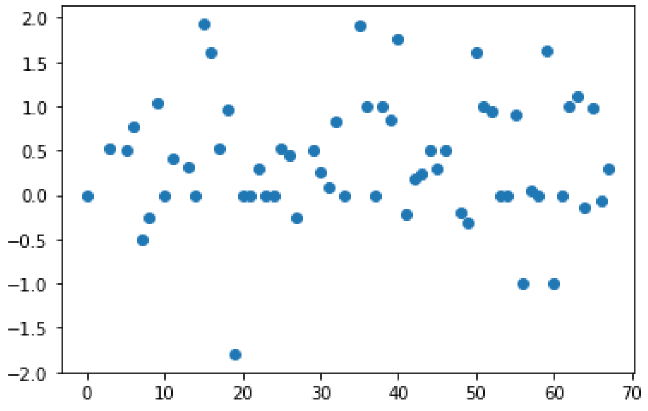
**Mean Sentiment(day) = (summation)no_of_retweets of tweet_i * sentiment of tweet_i / no of tweets for the day**

So after this procedure we can plot the day v/s mean sentiment response as x and y respectively.



Now the yellow points represents the outliers which again need to be removed. We again use the interquantile range 0.5 to 0.95.

After removing the outliers we get the points as follows,



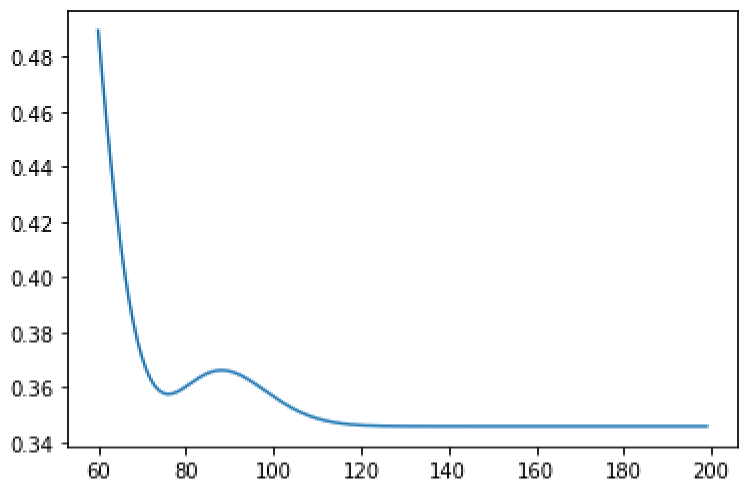
Now after getting all the mean sentiment for each day in the train sample we train the points using:

1) Linear Regression
2) Lasso Regression
3) Ridge Regression
4) Elastic Net Regression
5) Bayesian Ridge Regression
6) Decision Tree Regressor
7) Random Forest Regressor

We split the train dataset into 0.2 ratio as keep 20% of dataset as validation set.

| Model | R2 score of Validation | MSE of Validation Test | R2 Score of Train |
|---|---|---|---|
| Linear Regression | -0.0991516012015 | 0.587526556439 | 0.0010871567703 |
| Lasso Regression | -0.1022426148984 | 0.589178787698 | 0.0026043841252 |
| Ridge Regression | -0.1031674711573 | 0.589673148632 | 0.0026578440662 |
| Elastic Regression | -0.1026939284511 | 0.589420027120 | 0.0026444371327 |
| Bayesian Ridge | -0.0989561768916 | 0.587422096807 | 0.0008519670811 |
| SVR | -0.1403146740743 | 0.609529343344 | 0.0248225977940 |
| Decision Tree | -0.3265190834986 | 0.709060686740 | 0.4544485435374 |
| Random Forest | -0.1633043228977 | 0.621817938650 | 0.6131949366629 |

After analyzing the scores we use SVM as our predictor as the gap between R2 score of train and test is very less only bettered by linear,lasso and elastic regression.(So not much overfitting). But on the other hand R2 score of SVR for train is better than than linear,lasso and elastic Therefore model is not very underfitted to train as well.



The prediction of SVM for next 200 days (14-March 2020 is taken as day 0)

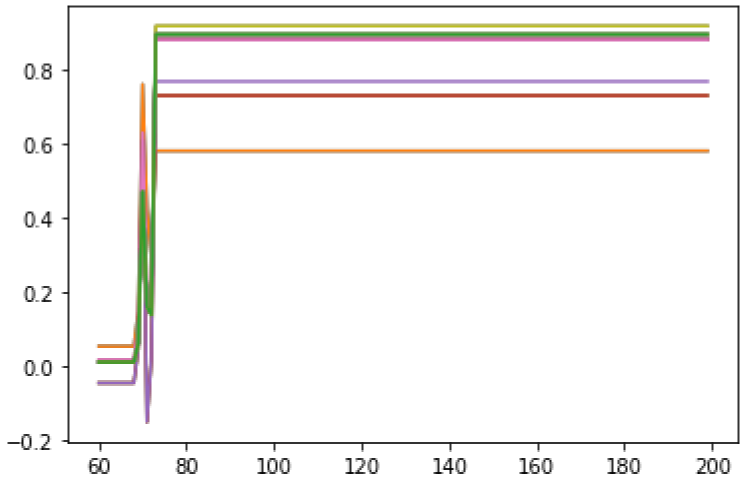
We can see that first positive sentiment increases and then gradually after a certain point of time people tend to obtain negative sentiment towards the lockdown and want it to be open.

**Statewise Mean Sentiment Prediction**
For this we have first seperated data based on states and stored the mean sentiments of a day in a particular state from the dataset.
Then we removed outliers just as we did before, from each state dataset.
Then we merged all the dataset of each state by adding another column of state index to input array.
Now we are training our model based on day and state and output is corresponding mean sentiment.

We split our dataset into 80-20 train and test dataset and trained our dataset on the same 7 models before.

| Model | R2 score of Train set | R2 score of Validation | MSE of validation |
|---|---|---|---|
| Linear Regression | 0.009891427009240063 | -0.0431419978318265955 | 0.0978735558761801 |
| Lasso Regression | 0.011055555462013533 | -0.04332907023727728 | 0.09789110808054521 |
| Ridge Regression | 0.013764294893562012 | -0.04127324374862029 | 0.09769821866651766 |
| Elastic Regression | 0.013084356742971814 | -0.041303057030624535 | 0.09770101591936355 |
| Bayesian Ridge | 0.01376429489341602 | -0.04127324260023979 | 0.09769821855877003 |
| SVR | 0.825881544359559 | -0.16169733987763865 | 0.1089970969839662 |
| Decision Tree | 0.5663426741599603 | -0.2117496111085868 | 0.1136932876993569 |
| Random Forest | 0.6708198466264456 | 0.016973617246401007 | 0.09223316461246031 |

We will use random forest as it has very less difference between the R2 score of train and validation samples and So there is less overfitting and also the R2 score of train is the best among them which means that it is not underfitted.



Prediction of RandomForest from day 60 -day 70.(14-March 2020 is taken as day 0,Different lines represent different states)

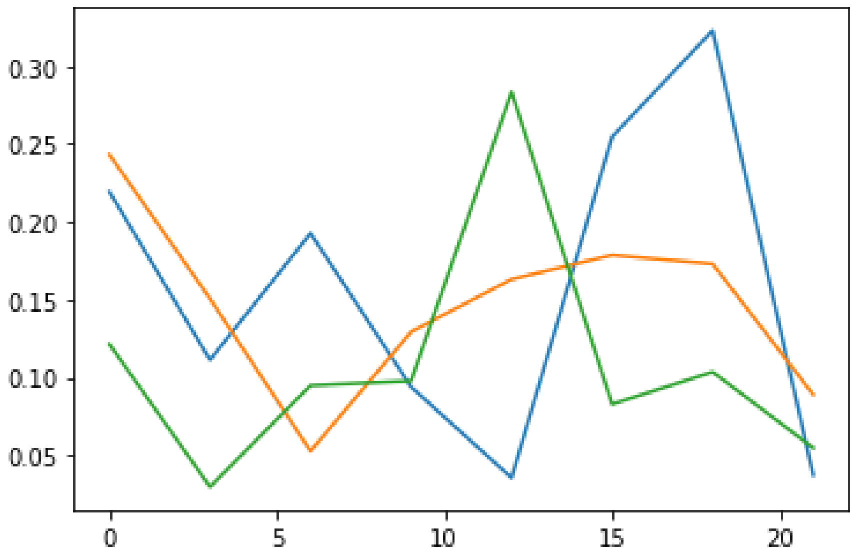
**Gender Wise Mean Sentiment**
For this we have first seperated data based on gender and stored the mean sentiments of a day of a particular gender from the dataset.
Then we removed outliers just as we did before, from each genderwise seperated dataset.
Then we merged all the dataset of each gender by adding another column of gender index to input array.
Now we are training our model based on day and gender and output is corresponding mean sentiment.

We split our dataset into 80-20 train and test dataset and trained our dataset on the same 7 models before.

| Model | R2 score of Validation | MSE of Validation | R2 score of Train |
|---|---|---|---|
| Linear Regression | 0.01086809533928 | 0.10163153010698 | 0.0013663847242 |
| Lasso Regression | -0.0102708605075 | 0.10157148479537 | 0.0012584975311 |
| Ridge Regression | 0.0556297230552 | 0.094945914977 | 0.0231853187702 |
| Elastic Regression | -0.012637725977 | 0.1018094467613 | 0.0013766551647 |
| Bayesian Ridge | 0.029834607621 | 0.0975393265836 | 0.0180032100592 |
| SVR | -0.10757326814 | 0.1113541583375 | -0.002619557095 |
| Decision Tree | -0.28906670841 | 0.1296013026728 | 0.455234052086 |
| Random Forest | -0.03721584133 | 0.1042805025625 | 0.494613149248 |

We will use random forest as it has very less difference between the R2 score of train and validation samples and So there is less overfitting and also the R2 score of train is the best among them which means that it is not underfitted.

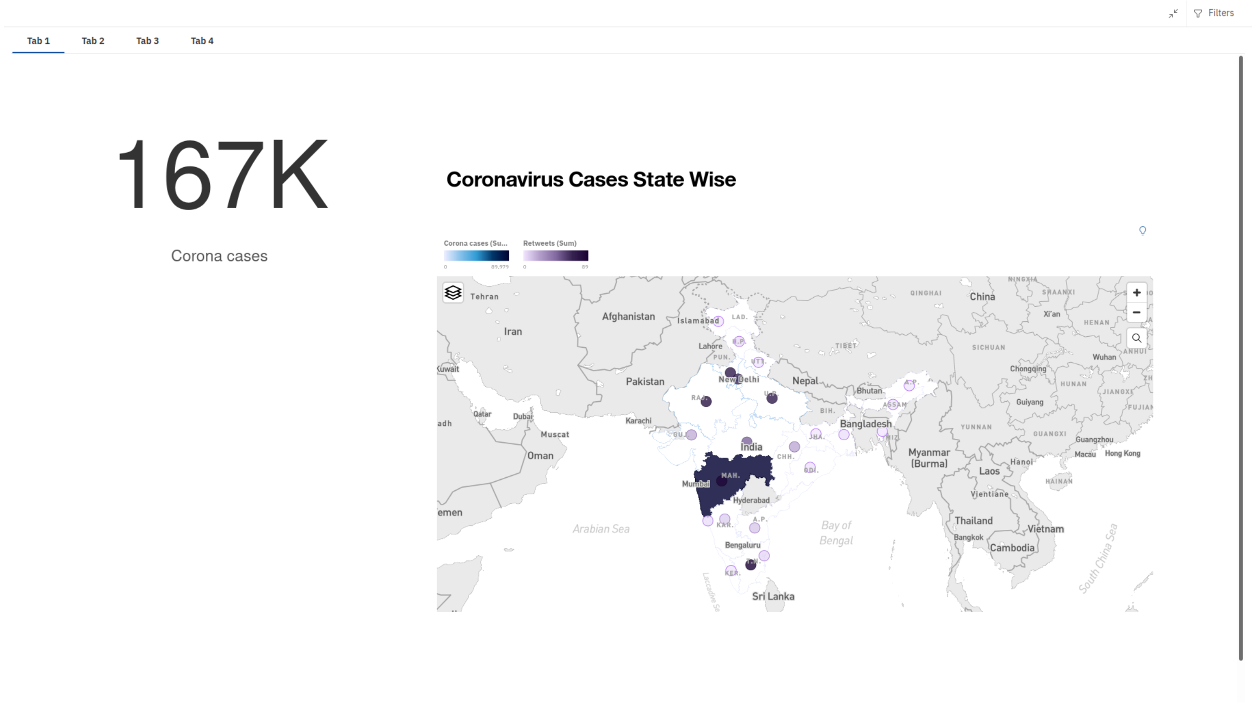
Prediction of next 20 days for different gender is represented by seperate lines



In the dashboard we have analyzed the data and also showed everthing on different different tabs.

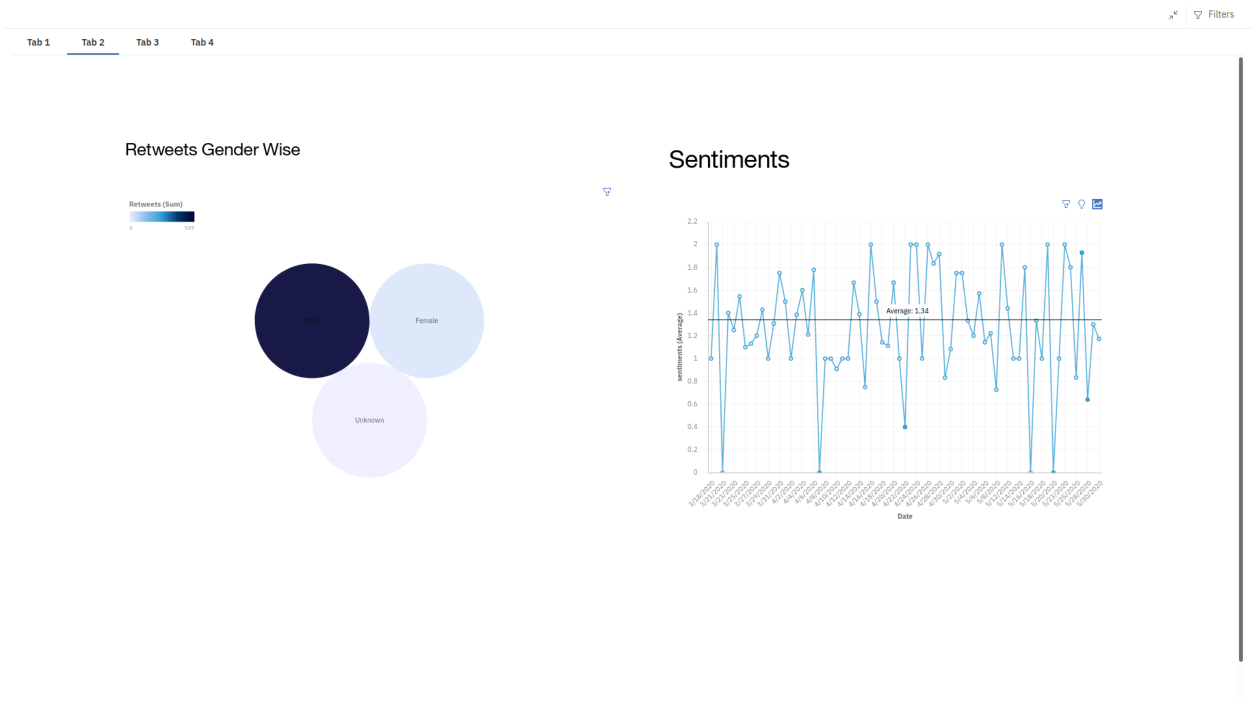
**Tab1**

It shows about the state wise distribution of the dataset.



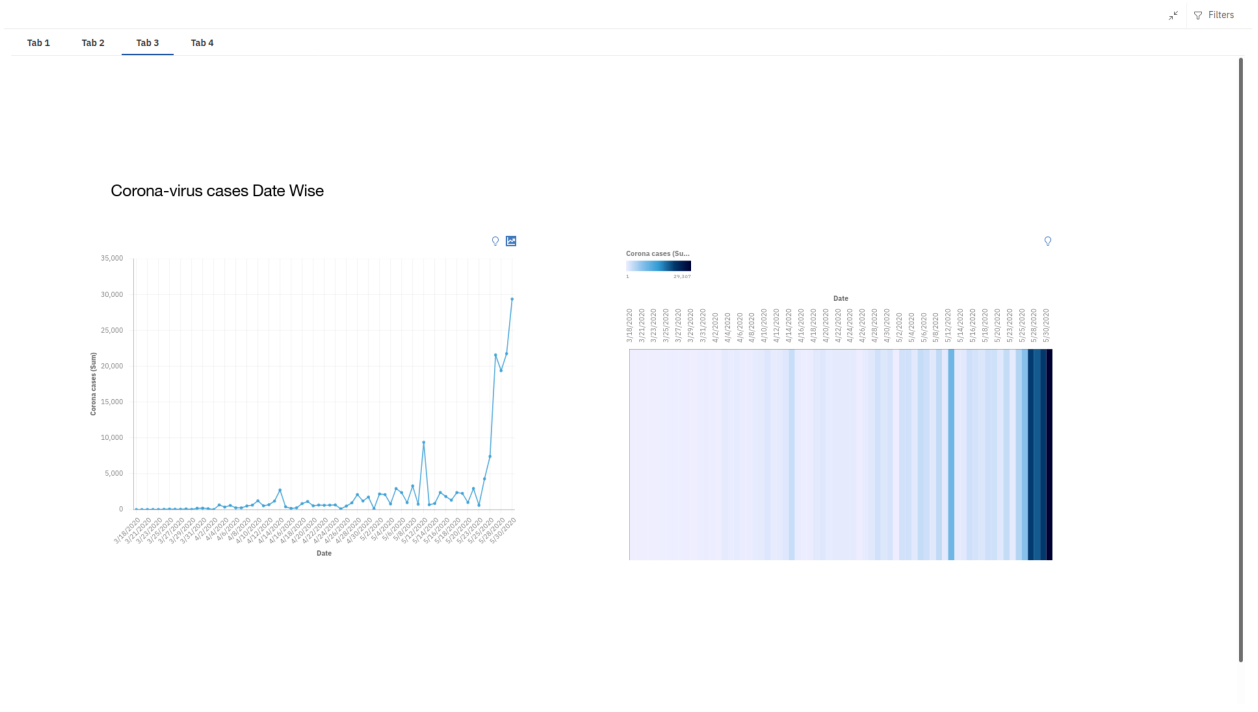
**Tab2**

It shows the distribution of data gender wise. And also the trend of sentiments of the past data.



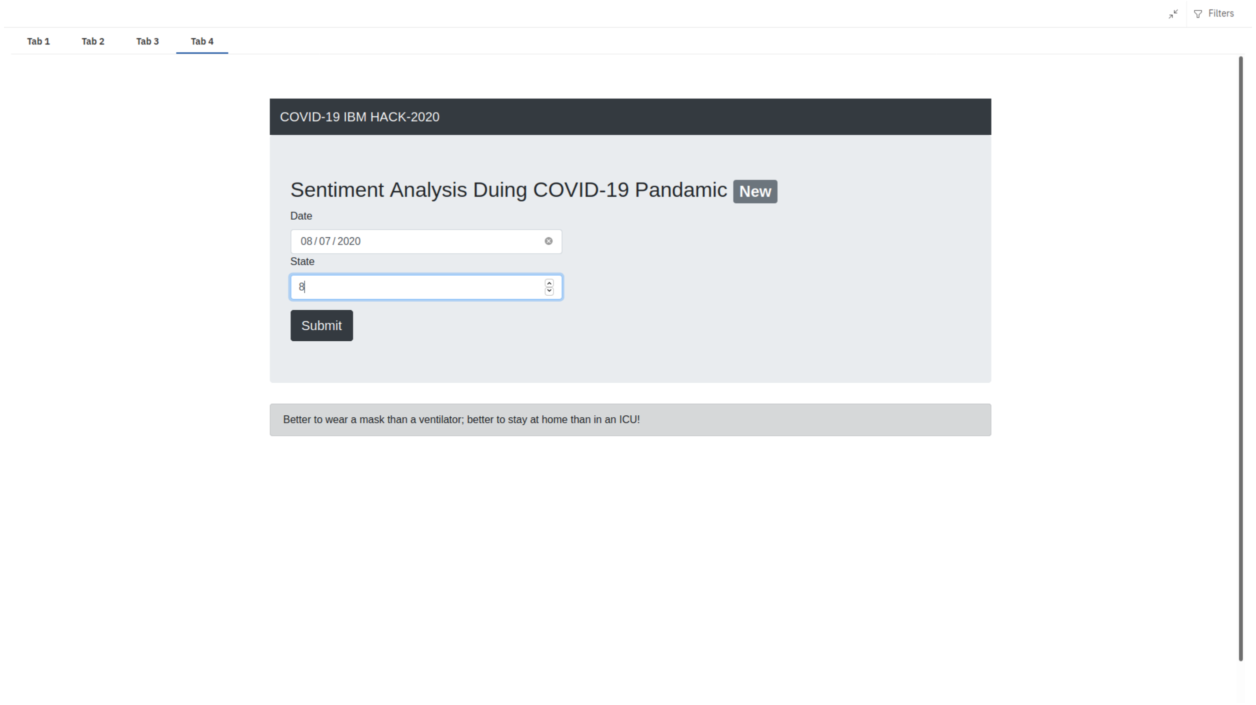
## Tab3

It shows the corona cases for everyday.



## Tab 4

It is an web app designed on Node.js and html to detect the sentiment in real time by taking two inputs from the user first the date and second the state code. And in our case we have mapped the state with value 0-29.



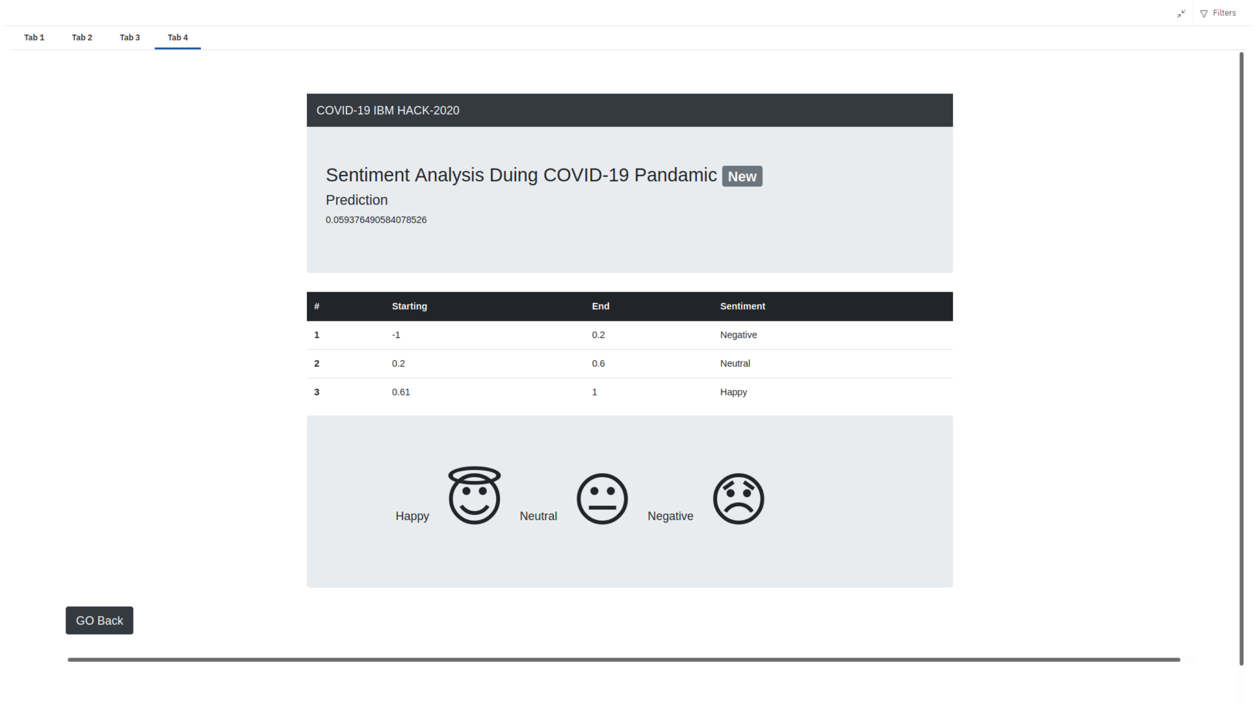
COVID-19 IBM HACK-2020

Sentiment Analysis Duing COVID-19 Pandamic New

Date: 08 / 07 / 2020

State: 8

Better to wear a mask than a ventilator; better to stay at home than in an ICU!



COVID-19 IBM HACK-2020

Sentiment Analysis Duing COVID-19 Pandamic New

Prediction

0.059376490584078526

| # | Starting | End | Sentiment |
|---|---|---|---|
| 1 | -1 | 0.2 | Negative |
| 2 | 0.2 | 0.6 | Neutral |
| 3 | 0.61 | 1 | Happy |