```python
# -*- coding: utf-8 -*-
"""
Created on Fri Sep 25 10:06:29 2020

@author: akash
"""

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from plotly.offline import init_notebook_mode, iplot
import matplotlib.pyplot as plt
import plotly.express as px
import plotly.graph_objects as go
import plotly.figure_factory as ff
from plotly.colors import n_colors
from plotly.subplots import make_subplots
import datetime
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
from sklearn.feature_extraction.text import CountVectorizer
from PIL import Image
from nltk.corpus import stopwords
stop=set(stopwords.words('english'))
from nltk.util import ngrams
import re
from collections import Counter
import nltk
from nltk.corpus import stopwords
import requests
import json
```

```
"""
Data Loading and Preperation
"""


tweets = pd.read_csv("C:/Users/akash/Desktop/S3/Exam/IPL2020_Tweets.csv")

tweets.shape

print(tweets['user_verified'].unique())

print(tweets['hashtags'].unique())

print(tweets['is_retweet'].unique())

#   Only one single value in 'is_retweet'
##  Hence we can delete that column

tweets = tweets.drop('is_retweet', axis=1)

tweets.shape

tweets[['date','time']] = tweets.date.str.split(expand=True)

tweets['hour'] = tweets.time.astype(str).str[:2]

def remove_tag(string):
    text=re.sub('<.*?>','',string)
    return text
```

```
def remove_mention(text):
    line=re.sub(r'@\w+','',text)
    return line


def remove_hash(text):
    line=re.sub(r'#\w+','',text)
    return line

def remove_newline(string):
    text=re.sub('\n','',string)
    return text


def remove_url(string):
    text = re.sub('http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[!*\(\),]|(?:%[0-9a-fA-F][0-9a-fA-
F]))+','',string)
    return text


def remove_number(text):
    line=re.sub(r'[0-9]+','',text)
    return line


def remove_punct(text):
    line = re.sub(r'[!"\$%&\'()*+,\-.\/:;=#@?\[\\\]^_`{|}~]*','',text)
    return line
```

```
def text_strip(string):
    line=re.sub('\s{2,}', ' ', string.strip())
    return line


def remove_thi_amp_ha_words(string):
    line=re.sub(r'\bamp\b|\bthi\b|\bha\b',' ',string)
    return line


tweets['refine_text']=tweets['text'].str.lower()

tweets['refine_text']=tweets['refine_text'].apply(lambda x:remove_tag(str(x)))

tweets['refine_text']=tweets['refine_text'].apply(lambda x:remove_mention(str(x)))

tweets['refine_text']=tweets['refine_text'].apply(lambda x:remove_hash(str(x)))

tweets['refine_text']=tweets['refine_text'].apply(lambda x:remove_newline(x))

tweets['refine_text']=tweets['refine_text'].apply(lambda x:remove_url(x))

tweets['refine_text']=tweets['refine_text'].apply(lambda x:remove_number(x))

tweets['refine_text']=tweets['refine_text'].apply(lambda x:remove_punct(x))

tweets['refine_text']=tweets['refine_text'].apply(lambda x:remove_thi_amp_ha_words(x))

tweets['refine_text']=tweets['refine_text'].apply(lambda x:text_strip(x))
```

```
tweets['text_length']=tweets['refine_text'].str.split().map(lambda x: len(x))

"""
Data Exploration

"""

##### Visualising date-wise number of tweets   {perfect}

tweets.date.value_counts().plot(kind='bar')

##### Year-wise creation of accounts     {perfect}

tweets['user_created'] = pd.to_datetime(tweets['user_created'])

tweets['year_created'] = tweets['user_created'].dt.year
```

```
df = tweets.drop_duplicates(subset='user_name', keep="first")

df = df[df['year_created']>2006]

df = df['year_created'].value_counts().reset_index()

df.columns = ['year', 'count']

fig = sns.barplot(x=df["year"], y=df["count"], orientation='vertical').set_title('Year-wise creation of users')

plt.xticks(rotation='vertical')

# We can see that majority of the accounts have been created in 2020.




######################## Top 10 locations based on number of tweets     {perfect}

locations = tweets['user_location'].value_counts().reset_index()

locations.columns = ['user_location', 'count']

locations = locations[locations['user_location']!='NA']

locations = locations.sort_values(['count'],ascending=False)

fig = sns.barplot(x=locations.head(10)["count"], y=locations.head(10)["user_location"], orientation='horizontal').set_title('Top 10 locations')

fig = sns.barplot(x=locations.head(11)["count"], y=locations.head(11)["user_location"], orientation='horizontal').set_title('Top 11 locations')
```

```
fig = sns.barplot(x=locations.head(16)["count"], y=locations.head(16)["user_location"], orientation='horizontal').set_title('Top 16 locations')
```

```
fig = sns.barplot(x=locations.head(22)["count"], y=locations.head(22)["user_location"], orientation='horizontal').set_title('Top 22 locations')
```

Now we get the top 10 locations as Mumbai, New Delhi, Hyderabad, Bengaluru, Chennai, Kolkata, Noida, Tamil Nadu, Jaipur and Kerala

Relation between number or tweets and number of hashtags per user {perfect}

```
tweets['hashtags'] = tweets['hashtags'].fillna('[]')
```

```
tweets['hashtags_count'] = tweets['hashtags'].apply(lambda x: len(x.split(',')))
```

```
tweets.loc[tweets['hashtags'] == '[]', 'hashtags_count'] = 0
```

```
tweets['hashtags_count']
```

```
userwise = tweets['user_name'].value_counts().reset_index()
```

```
userwise.columns = ['user_name', 'tweets_count']
```

```
fig = sns.scatterplot(x=tweets['hashtags_count'], y=userwise['tweets_count']).set_title('Number of hashtags used in Tweets')
```

```
## We can see that number of tweets is inversely proportional to the number of hashtags in a tweet

#### Hourly count of tweets  {perfect}

hourly = tweets['hour'].value_counts().reset_index()

hourly.columns = ['hour', 'count']

hourly['hour'] = 'Hour ' + hourly['hour'].astype(str)

fig = sns.barplot(x=hourly["hour"], y=hourly["count"], orientation='vertical',).set_title('Tweets distribution over hours')

plt.xticks(rotation='vertical')

# We can observe that most tweets are tweeted around afternoon 1 (hour 13) until 6 PM (hour 18)
```

```
#### Hashtags per tweet

htcounts = tweets['hashtags_count'].value_counts().reset_index()

htcounts.columns = ['hashtags_count', 'count']

htcounts = htcounts.sort_values(['count'],ascending=False)

fig = sns.barplot(x=htcounts["hashtags_count"], y=htcounts["count"],
orientation='vertical').set_title('Number of hashtags per tweet')

# We can confirm our previous observation of inverse proportion between hashtags and tweet
counts

########## Top 10 hashtags used in tweets

hts = tweets['hashtags'].value_counts().reset_index()

hts.columns = ['hashtag', 'count']

hts = hts.sort_values(['count'],ascending=False)

fig = sns.barplot(x=hts.head(10)['hashtag'],
y=hts.head(10)['count'],orientation='vertical').set_title('Top 10 hashtags')
plt.xticks(rotation = 'vertical')

fig = sns.barplot(x=hts.head(17)['hashtag'],
y=hts.head(17)['count'],orientation='vertical').set_title('Top 17 hashtags')
plt.xticks(rotation = 'vertical')
```

```
# Hence top 10 hashtags are #IPL2020/IPL, #DCvKXIP, #CSKvsRR (and vice versa), #CSK

# #IPLwithNews18, #RCBvsKXIP, #RCB, #SRHvsRCB, #MIvsCSK


# All matches except KKR v/s MI appeared in the top 10 tweets


"""

Analysis


"""


###### Splitting the data into sets of Tweets that were tweeted before the match date (19/09/2020) and after the matches started


tweets['date'] = pd.to_datetime(tweets['date'], yearfirst=(True))


tweets['date'].unique()
```

```
prematch = tweets[(tweets['date'] >= '2020-08-15') & (tweets['date'] < '2020-09-19')]

prematch['date'].unique()

prematch.shape
## pre-match tweets constitutes of 39.01% of the data



matchdays = tweets[(tweets['date'] > '2020-09-18') & (tweets['date'] <= '2020-09-24')]

matchdays['date'].unique()

matchdays.shape
## and tweets during matches constitue of the rest 60.99% of the data.



############## Pre-Match tweets analysis



######## Hourly analysis
```

```
hour = prematch['hour'].value_counts().reset_index()

hour.columns = ['hour', 'count']

fig = sns.barplot(x=hour["hour"], y=hour['count'],orientation='vertical').set_title('Pre-Match Number of Tweets on hourly basis')
# We can assume that this follows a normal distribution



##########   Wordcloud



text = prematch['text']

text = str(text)

wordcloud = WordCloud(max_font_size=50, max_words=100, background_color="white").generate(text)
plt.figure()
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.show()



########### N-gram
```

```
def ngram_df(corpus,nrange,n=None):
    vec = CountVectorizer(stop_words = 'english',ngram_range=nrange).fit(corpus)
    bag_of_words = vec.transform(corpus)
    sum_words = bag_of_words.sum(axis=0)
    words_freq = [(word, sum_words[0, idx]) for word, idx in vec.vocabulary_.items()]
    words_freq =sorted(words_freq, key = lambda x: x[1], reverse=True)
    total_list=words_freq[:n]
    df=pd.DataFrame(total_list,columns=['text','count'])
    return df


unigram_pre=ngram_df(prematch['refine_text'],(1,1),10)


bigram_pre=ngram_df(prematch['refine_text'],(2,2),10)


trigram_pre=ngram_df(prematch['refine_text'],(3,3),10)


fig = sns.barplot(x=unigram_pre["text"], y=unigram_pre['count'],orientation='vertical').set_title('Top 10 single words')
plt.xticks(rotation='vertical')


fig = sns.barplot(x=bigram_pre['text'], y=bigram_pre['count'],orientation='vertical').set_title('Top 10 double words')
plt.xticks(rotation='vertical')


fig = sns.barplot(x=trigram_pre['text'], y=trigram_pre['count'],orientation='vertical').set_title('Top 10 triple words')
plt.xticks(rotation='vertical')


#    We can observe that most pre-match tweets are are regarding teams and sponsors of the events
```

```
##   A certain team is highly mentioned because of the players that made the news a around this timeline

###############  Analysis of tweets that were tweeted on match days

######## Hourly analysis

hour = matchdays['hour'].value_counts().reset_index()

hour.columns = ['hour', 'count']

fig = sns.barplot(x=hour["hour"], y=hour['count'],orientation='vertical').set_title('Number of Tweets on hourly basis on Match days')

##########   Wordcloud

text = matchdays['text']
```

```
text = str(text)

wordcloud = WordCloud(max_font_size=50, max_words=100,
background_color="white").generate(text)

plt.figure()

plt.imshow(wordcloud, interpolation="bilinear")

plt.axis("off")

plt.show()



########### N-gram



unigram_match=ngram_df(matchdays['refine_text'],(1,1),10)

bigram_match=ngram_df(matchdays['refine_text'],(2,2),10)

trigram_match=ngram_df(matchdays['refine_text'],(3,3),10)

fig = sns.barplot(x=unigram_match["text"],
y=unigram_match['count'],orientation='vertical').set_title('Top 10 single words')

plt.xticks(rotation='vertical')

fig = sns.barplot(x=bigram_match['text'],
y=bigram_match['count'],orientation='vertical').set_title('Top 10 double words')

plt.xticks(rotation='vertical')
```

```
fig = sns.barplot(x=trigram_match['text'],
y=trigram_match['count'],orientation='vertical').set_title('Top 10 triple words')

plt.xticks(rotation='vertical')
```

```
########### Diving deeper into match day Tweets
```

```
################################# Day 1 - 19/09/2020: MI v/s CSK = CSK won
```

```
day1 = tweets[(tweets['date'] > '2020-09-18') & (tweets['date'] <= '2020-09-19')]

day1['date'].unique()
```

```
######## Hourly analysis
```

```
hour = day1['hour'].value_counts().reset_index()

hour.columns = ['hour', 'count']

fig = sns.barplot(x=hour["hour"], y=hour['count'],orientation='vertical').set_title('Number of Tweets on hourly basis on Day 1 of schedule')

##########   Wordcloud

text = day1['text']

text = str(text)

wordcloud = WordCloud(max_font_size=50, max_words=100, background_color="white").generate(text)
plt.figure()
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.show()

################################ Day 2 - 20/09/2020: DC v/s KXIP = DC won in super over
```

```
day2 = tweets[(tweets['date'] > '2020-09-19') & (tweets['date'] <= '2020-09-20')]

day2['date'].unique()

######## Hourly analysis

hour = day2['hour'].value_counts().reset_index()

hour.columns = ['hour', 'count']

fig = sns.barplot(x=hour["hour"], y=hour['count'],orientation='vertical').set_title('Number of Tweets on hourly basis on Day 2 of schedule')

##########   Wordcloud

text = day2['text']

text = str(text)

wordcloud = WordCloud(max_font_size=50, max_words=100, background_color="white").generate(text)
```

```
plt.figure()
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.show()



################################ Day 3 - 21/09/2020: SRH v/s RCB =  RCB won



day3 = tweets[(tweets['date'] > '2020-09-20') & (tweets['date'] <= '2020-09-21')]
day3['date'].unique()



######## Hourly analysis



hour = day3['hour'].value_counts().reset_index()

hour.columns = ['hour', 'count']

fig = sns.barplot(x=hour["hour"], y=hour['count'],orientation='vertical').set_title('Number of Tweets on hourly basis on Day 3 of schedule')
```

```
##########   Wordcloud

text = day3['text']

text = str(text)

wordcloud = WordCloud(max_font_size=50, max_words=100,
background_color="white").generate(text)
plt.figure()
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.show()

################################ Day 4 - 22/09/2020: RR v/s CSK =  RR won

day4 = tweets[(tweets['date'] > '2020-09-21') & (tweets['date'] <= '2020-09-22')]
day4['date'].unique()

######## Hourly analysis
```

```
hour = day4['hour'].value_counts().reset_index()

hour.columns = ['hour', 'count']

fig = sns.barplot(x=hour["hour"], y=hour['count'],orientation='vertical').set_title('Number of Tweets on hourly basis on Day 4 of schedule')

##########   Wordcloud

text = day4['text']

text = str(text)

wordcloud = WordCloud(max_font_size=50, max_words=100, background_color="white").generate(text)
plt.figure()
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.show()

################################ Day 5 - 23/09/2020: KKR v/s MI =  MI won
```

```
day5 = tweets[(tweets['date'] > '2020-09-22') & (tweets['date'] <= '2020-09-23')]

day5['date'].unique()

######## Hourly analysis

hour = day5['hour'].value_counts().reset_index()

hour.columns = ['hour', 'count']

fig = sns.barplot(x=hour["hour"], y=hour['count'],orientation='vertical').set_title('Number of Tweets on hourly basis on Day 5 of schedule')

##########   Wordcloud

text = day5['text']

text = str(text)
```

```
wordcloud = WordCloud(max_font_size=50, max_words=100,
background_color="white").generate(text)

plt.figure()

plt.imshow(wordcloud, interpolation="bilinear")

plt.axis("off")

plt.show()




################################ Day 6 - 24/09/2020: KXIP v/s RCB =  KXIP won by 97 runs




day6 = tweets[(tweets['date'] > '2020-09-23') & (tweets['date'] <= '2020-09-24')]

day6['date'].unique()




######## Hourly analysis




hour = day6['hour'].value_counts().reset_index()


hour.columns = ['hour', 'count']


fig = sns.barplot(x=hour["hour"], y=hour['count'],orientation='vertical').set_title('Number of Tweets
on hourly basis on Day 6 of schedule')
```

```
##########   Wordcloud

text = day6['text']

text = str(text)

wordcloud = WordCloud(max_font_size=50, max_words=100,
background_color="white").generate(text)

plt.figure()

plt.imshow(wordcloud, interpolation="bilinear")

plt.axis("off")

plt.show()
```

```
#      From daily analysis of match day data, we can concur that tweets reduce significantly after the match starts until the match ends.

##     Because of this, we can also observe a some amount of tweets regarding previous day's game in present day's analysis.

###    Looking at daily tweet analysis in Data Exploration and match-day analysis, we can concur that high-adrenaline events from the match might be high contributors to the spike in number of tweets

####   More events such as high scores, dropped catches, excpetional fieldings may also contribute hugely to the spike.

#####  We can also observe that whenever CSK is in game, words such as 'dhoni', 'msd', 'msdian', 'thala', etc are always mentioned in high numbers.

###### This has changed from pre-match analysis where in 'Suresh Raina' was bieng mentioned more.
```

#######Also it can be observed that match day tweets are majorly dominated by high-adrenaline events, certain high performing player, captain of the winning team and match polls.