**Example Basic Python Machine Learning Code**

Thank you for listening to my presentation today, for any questions please contact me at shohei.kato@trader.ca

**Demo 1:**

**Extract comments from a RedFlagDeal forum discussing Instant Cash Offer, and create wordmap**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

#import the needed libraries

from bs4 import BeautifulSoup

import requests

from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator

import matplotlib.pyplot as plt

import matplotlib.image as mpimg

#-----------------------------------------------------------

#Get website data

forum = requests.get("http://forums.redflagdeals.com/buying-used-new-car-cash-discount-2254637/")

#extract just comments and parse data into frame

soup = BeautifulSoup(forum.content, 'html.parser')

comments1 = " ".join([i.text for i in soup.find\_all('div', class\_='content')])

#-----------------------------------------------------------

#create a word cloud

wordcloud = WordCloud(max\_font\_size=80, max\_words=200, background\_color='white').generate(comments1)

# Display the generated image:

plt.figure(figsize=(12,6)) #set size first

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

plt.axis("off")

plt.savefig('image.png',dpi = 100)

plt.show()

#If packages aren’t imported properly, you might need to upgrade pip and install wordcloud

import sys

!{sys.executable} -m pip install --upgrade pip

!{sys.executable} -m pip install wordcloud

**Demo 2:**

**Extract comments with Canadian International Autoshow 2019 hashtag (#LeBronjames) and conduct sentiment analysis**

* You will need to obtain Twitter API keys to obtain this analysis, once you obtain

#------------------------------------------------------------

First we will store Twitter API credentials in a separate file for safety

import json

# create a dictionary to store your twitter credentials

credentials = dict()

# Enter your own consumer\_key, consumer\_secret, access\_key and access\_secret

# Replacing the stars ("\*\*\*\*\*\*\*\*")

# Consume:

credentials['CONSUMER\_KEY'] = '\*\*\*\*\*\*\*\*'

credentials['CONSUMER\_SECRET'] = '\*\*\*\*\*\*\*\*'

# Access:

credentials['ACCESS\_TOKEN'] = '\*\*\*\*\*\*\*\*'

credentials['ACCESS\_SECRET'] = '\*\*\*\*\*\*\*\*'

with open('credentials.json', 'w') as secret\_info:

json.dump(credentials, secret\_info, indent=4, sort\_keys=True)

#------------------------------------------------------------

Open a separate Jupyter notebook, and import packages, call out Twitter APIs

import tweepy

import csv

import pandas as pd

import numpy as np # For number computing

import json

import re

# You will need to set up Twitter API

with open('twitter\_credentials.json') as cred\_data:

info = json.load(cred\_data)

CONSUMER\_KEY = info['CONSUMER\_KEY']

CONSUMER\_SECRET = info['CONSUMER\_SECRET']

ACCESS\_TOKEN = info['ACCESS\_TOKEN']

ACCESS\_SECRET = info['ACCESS\_SECRET']

auth = tweepy.OAuthHandler(CONSUMER\_KEY, CONSUMER\_SECRET)

auth.set\_access\_token(ACCESS\_TOKEN, ACCESS\_SECRET)

api = tweepy.API(auth)

# Search API limits us to 100 tweets per query

# We want to switch to application-only authentication for the higher rate limit.

# Tweepy has a separate AppAuthHandler method for this purpose:

#Switching to application authentication

auth = tweepy.AppAuthHandler(CONSUMER\_KEY, CONSUMER\_SECRET)

#Setting up new api wrapper, using authentication only

api = tweepy.API(auth, wait\_on\_rate\_limit=True,wait\_on\_rate\_limit\_notify=True)

#You can check how many queries you have left using rate\_limit\_status() method (remove #)

api.rate\_limit\_status()['resources']['search']

#search term

searchQuery = '#LBJ OR #LeBronJames OR #Lebron OR #Lebronjames'

#searchQuery = '#cias2019'

#Maximum number of tweets we want to collect

maxTweets = 1000000

#The twitter Search API allows up to 100 tweets per query

tweetsPerQry = 100

tweets = []

for tweet in tweepy.Cursor(api.search,q=searchQuery,show\_user = True,lang='en').items(100):

tweets.append(tweet)

for tweet in tweets:

print(tweet.text.encode('utf-8'))

#Code below tells you what kind of parameters you can pull

print(dir(tweets[0]))

#We will insert fetched data into Dataframe

#df['just\_date'] = df['dates'].dt.date

#del data

def insertDf(tweets):

data = pd.DataFrame()

data['ID'] = [tweet.id for tweet in tweets]

data['userID'] = [tweet.user.id for tweet in tweets]

data['userScreen'] = [tweet.user.screen\_name for tweet in tweets]

data['userName'] = [tweet.user.name for tweet in tweets]

data['Date']= [tweet.created\_at for tweet in tweets]

data['Text'] = [tweet.text for tweet in tweets]

data['clean\_tweet'] = [tweet.text for tweet in tweets]

data['Retweet Cnt'] = [tweet.retweet\_count for tweet in tweets]

data['Tweet Fav'] = [tweet.favorite\_count for tweet in tweets]

data['Source'] = [tweet.source for tweet in tweets]

return data

#Pass the tweets list to the above function to create a DataFrame

data = insertDf(tweets)

#Let’s check how the table looks like

data

#We’ll clean up the tweets for better accuracy of sentiment analysis

data['clean\_tweet'] = data['clean\_tweet'].str.replace("@[\w]\*","")

data['clean\_tweet'] = data['clean\_tweet'].str.replace('https:\S+|www.\S+|http:\S+', '')

data['clean\_tweet'] = data['clean\_tweet'].str.replace("[^0-9A-Za-z \t \#]", "")

#Now we are introducing Textblob package. Textblob package calculates sentiment and subjectivity based on the input

from textblob import TextBlob

TextBlob("TTC is delayed").sentiment

#We will categorize each tweets into “Positive”, ”Neutral”, and ”Negative” brackets

tweet = data['clean\_tweet']

def analize\_sentiment(tweet):

'''

Utility function to classify the polarity of a tweet

using textblob.

'''

analysis = TextBlob(tweet)

if analysis.sentiment.polarity > 0:

return 1

elif analysis.sentiment.polarity == 0:

return 0

else:

return -1

# We create a column with the result of the analysis:

data['SA'] = np.array([ analize\_sentiment(tweet) for tweet in data['clean\_tweet'] ])

# We display the updated dataframe with the new column:

display(data)

# We construct lists with classified tweets:

pos\_tweets = [ tweet for index, tweet in enumerate(data['clean\_tweet']) if data['SA'][index] > 0]

neu\_tweets = [ tweet for index, tweet in enumerate(data['clean\_tweet']) if data['SA'][index] == 0]

neg\_tweets = [ tweet for index, tweet in enumerate(data['clean\_tweet']) if data['SA'][index] < 0]

# We print percentages:

print(f"Percentage of positive tweets: {len(pos\_tweets)\*100/len(data['clean\_tweet'])}%")

print(f"Percentage of neutral tweets: {len(neu\_tweets)\*100/len(data['clean\_tweet'])}%")

print(f"Percentage de negative tweets: {len(neg\_tweets)\*100/len(data['clean\_tweet'])}%")

# Let’s export the result

data.to\_csv('example.csv')