WhatsApp Chat Analyser

Howdy! So, lets first talk about the utility of
this project and how will we do that.

The basic idea behind this project is to generate API which can store all our prediction in MongoDB.

What we do Today

Let's go through the quick over view before starting the project.
Path we follow:

- Collection of training dataset
- Pre-processing
- Applying Model
- Setup pipeline and save it
- Export WhatsApp chat
- Building API and Storing prediction in MongoDB

Collection of Dataset

The basic idea of this blog is to give you the proper pathway so I had not taken large data as it will consume more time for training and All but you can train it with larger data set for practical use.
Either manually or by Web Scrapping collect interview questions related to ML, Bigdata, and ReactJs then save it to csv files as ML_interview.csv, BigData_interview.csv and Reactjs_interview.csv. I had collected data from the following links:
https://www.guru99.com/machine-learning-interview-questions.html
https://www.whizlabs.com/blog/big-data-interview-questions/
https://mindmajix.com/reactjs-interview-questions

And if you want to use my data then download it from below links:
https://docs.google.com/spreadsheets/d/e/2PACX-1vSnX87hWa_p2PkUpN5ltlv82M-DNL2O6508XuWF4ReVc4UVVWMIHdbepm7Lrpyld_joyYlAQA3cjPbh/pub?gid=1394236440&single=true&output=csv
https://docs.google.com/spreadsheets/d/e/2PACX-1vRaJGkyoUdt5Zm37tUVK7dUgTxxuRt2yHbpjFCOZzTOIRmPlHpP7-8el5ZjjDw22YU5CtbdKhcRN4B5/pub?gid=404467716&single=true&output=csv
https://docs.google.com/spreadsheets/d/e/2PACX-1vTimX84Wfq2pW_qAeEi4XUXQH74BBwtsVKaJYeS7ZsTLG700SVzKS3

qEMWCjZ3ePyh-4l6I3vGpXNO0/pubhtml?gid=667577516&single=true

First import libraries:

```
import pandas as pd
import numpy as np
```

Now append the all the data frames that holding the individual csv files:

```
def Prep(label):
    x=[]
    for i in range(50):
        x.append(label)
    return x
```

```
df1 = pd.read_csv('ML_interview.csv')
df1['Label']=Prep('ML')
df2 = pd.read_csv("Reactjs_interview.csv")
df2['Label']=Prep('ReactJs')
df3 = pd.read_csv("BigData_interview.csv")
df3['Label']=Prep('BigData')
```

```
train=df1.append(df2).append(df3)
train.drop('Unnamed: 0',1,inplace=True)
train=train.fillna(' ')
```

```
In [6]: train['Label'].unique()
```

```
Out[6]: array(['ML', 'ReactJs', 'BigData'], dtype=object)
```

So, now our data frame is ready for pre-processing.

Pre-processing

First import all the useful packages:

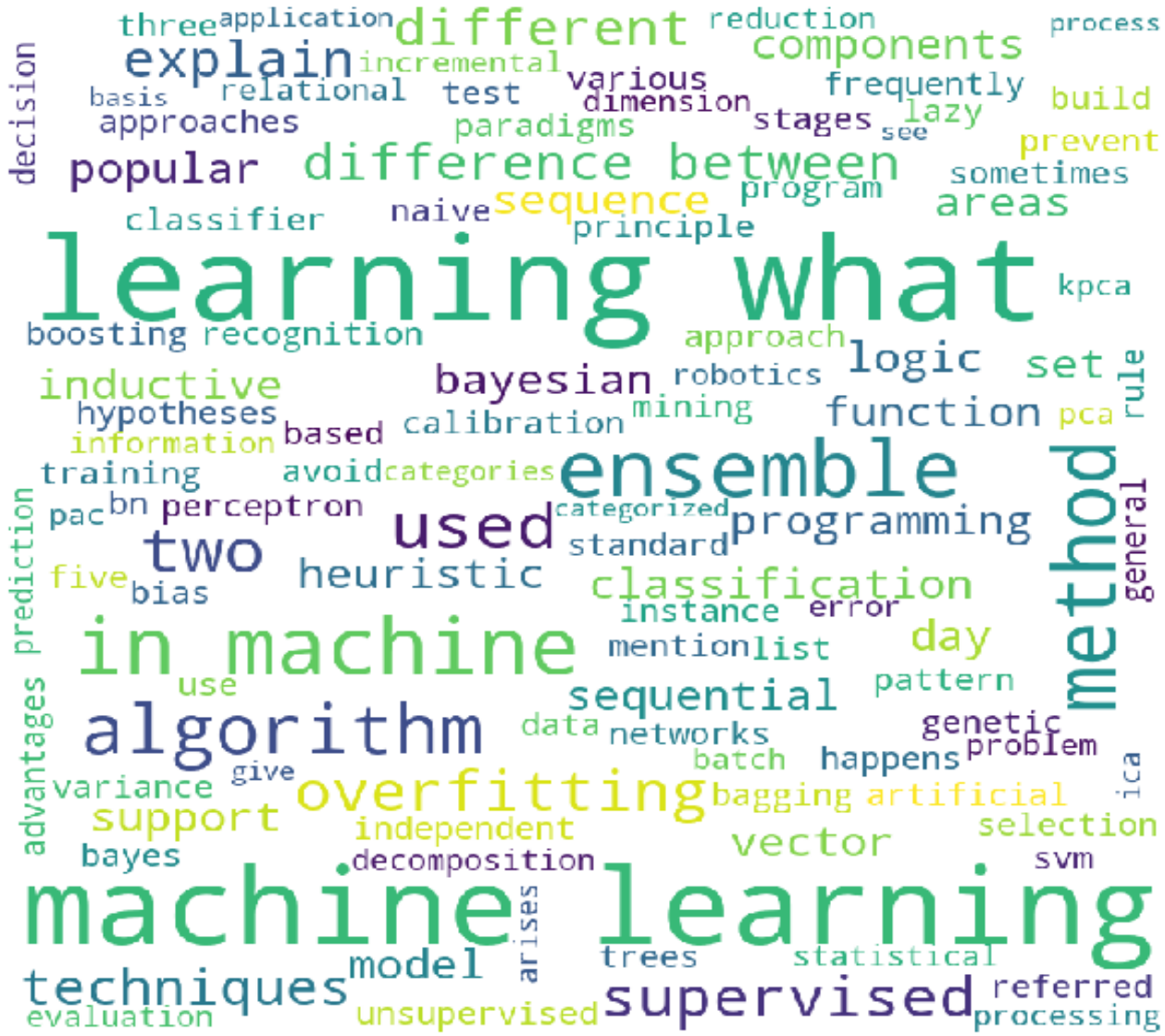
```
from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt
from sklearn import metrics
from sklearn.metrics import confusion_matrix
%matplotlib inline
import seaborn as sns

import numpy as np # linear algebra
import pandas as pd #data processing

import os
import re
import nltk
```

Now form a cloud visual to familiarize more with your dataset.

```
real_words = ''
fake_words = ''
stopwords = set(STOPWORDS)

# iterate through the csv file
for val in train[train['Label']==0].questions:

    # split the value
    tokens = val.split()

    # Converts each token into lowercase
    for i in range(len(tokens)):
        tokens[i] = tokens[i].lower()

    real_words += " ".join(tokens)+" "

for val in train[train['Label']==1].questions:

    # split the value
    tokens = val.split()

    # Converts each token into lowercase
    for i in range(len(tokens)):
        tokens[i] = tokens[i].lower()

    fake_words += " ".join(tokens)+" "

for val in train[train['Label']==2].questions:

    # split the value
    tokens = val.split()

    # Converts each token into lowercase
    for i in range(len(tokens)):
        tokens[i] = tokens[i].lower()

    fake_words += " ".join(tokens)+" "
```

```
wordcloud = WordCloud(width = 800, height = 800,
                background_color ='white',
                stopwords = stopwords,
                min_font_size = 10).generate(real_words)

# plot the WordCloud image
plt.figure(figsize = (8, 8), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)

plt.show()
```

Then a beautiful image come as output of the above code which is actually our cloud visual.



Now let's apply NLP techniques.

```
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords
lemmatizer=WordNetLemmatizer()
nltk.download('stopwords')
stop_words = stopwords.words('english')
lemmatizer=WordNetLemmatizer()
for index,row in train.iterrows():
    filter_sentence = ''

    sentence = row['questions']
    sentence = re.sub(r'[^\w\s]','',sentence) #cleaning

    words = nltk.word_tokenize(sentence) #tokenization

    words = [w for w in words if not w in stop_words]  #stopwords removal

    for word in words:
        filter_sentence = filter_sentence + ' ' + str(lemmatizer.lemmatize(word)).lower()

    train.loc[index,'questions'] = filter_sentence

train = train[['questions','Label']]
```

Now we have a clean dataset
Let's check how our dataset looks like now.

```
In [113]: train.head()
```

Out[113]:

	questions	Label
0	What is Machine learning?	0
1	Mention the difference between Data Mining an...	0
2	What is 'Overfitting' in Machine learning?	0
3	Why overfitting happens?	0
4	How can you avoid overfitting ?	0

Prepare a function for Tf-idf vectorizer.

In [114]:
```
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [115]:
```
X_train=train['questions']
Y_train=train['Label']
```

TF-iDF Vectorizer

In [116]:
```
def vectorize_text(features, max_features):
    vectorizer = TfidfVectorizer( stop_words='english',
                                  decode_error='strict',
                                  analyzer='word',
                                  ngram_range=(1, 2),
                                  max_features=max_features
                                  #max_df=0.5 # Verwendet im ML-Kurs unter Preprocessing
                                  )
    feature_vec = vectorizer.fit_transform(features)
    return feature_vec.toarray()
```

Tf-idf transform the text into the feature vectors and used as input to the estimator.

Apply

In [117]:
```
#Feature extraction using count vectorization and tfidf.
count_vectorizer = CountVectorizer()
count_vectorizer.fit_transform(X_train)
freq_term_matrix = count_vectorizer.transform(X_train)
tfidf = TfidfTransformer(norm="l2")
tfidf.fit(freq_term_matrix)
tf_idf_matrix = tfidf.fit_transform(freq_term_matrix)
```

In [118]:
```
tf_idf_matrix
```

Out[118]:
```
<150x374 sparse matrix of type '<class 'numpy.float64'>'
        with 1182 stored elements in Compressed Sparse Row format>
```

Congratulations your text is successfully stored in tf_idf matrix. It is now fit for applying Model.

Applying Model

Let's train our data using ***Logistic Regression, MultinomialNB, and Random Forest Classifier.***

Logistic Regression:

In [36]:
```
#split in samples
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(tf_idf_matrix, Y_train,test_size=0.3,\
                                                    random_state=0)
#Logistic Regression

from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression(C=1e5)
logreg.fit(X_train, y_train)
pred = logreg.predict(X_test)
print('Accuracy of Lasso classifier on training set: {:.2f}'
     .format(logreg.score(X_train, y_train)))
print('Accuracy of Lasso classifier on test set: {:.2f}'
     .format(logreg.score(X_test, y_test)))
```

```
Accuracy of Lasso classifier on training set: 1.00
Accuracy of Lasso classifier on test set: 0.93
```

In [37]:
```
from sklearn.metrics import f1_score,recall_score,precision_score
print(f1_score(y_test,pred,average='micro'))
print(recall_score(y_test,pred,average='micro'))
print(precision_score(y_test,pred,average='micro'))
```

```
0.9333333333333333
0.9333333333333333
0.9333333333333333
```

MultinomialNB:

In [76]:
```
#split in samples
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(tf_idf_matrix, Y_train,\
                                                    test_size=0.3,\
                                                    random_state=0)

from sklearn.naive_bayes import MultinomialNB

NB = MultinomialNB()
NB.fit(X_train, y_train)
pred = NB.predict(X_test)
print('Accuracy of NB  classifier on training set: {:.2f}'
     .format(NB.score(X_train, y_train)))
print('Accuracy of NB classifier on test set: {:.2f}'
     .format(NB.score(X_test, y_test)))
```

```
Accuracy of NB  classifier on training set: 1.00
Accuracy of NB classifier on test set: 0.78
```

In [77]:
```
from sklearn.metrics import f1_score,recall_score,pre
print(f1_score(y_test,pred,average='micro'))
print(recall_score(y_test,pred,average='micro'))
print(precision_score(y_test,pred,average='micro'))
```

```
0.7777777777777778
0.7777777777777778
0.7777777777777778
```

Random Forest Classifier:

In [78]:
```
#split in samples
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(tf_idf_matrix, Y_train,test_size=0.3,\
                                                    random_state=0)

from sklearn.metrics import accuracy_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import make_classification
X, y = make_classification(n_samples=1000, n_features=4,
                           n_informative=2, n_redundant=0,
                           random_state=0, shuffle=False)
clf = RandomForestClassifier(max_depth=9, random_state=0)
clf.fit(X_train, y_train)
pred=clf.predict(X_test)
print('Accuracy of RF  classifier on training set: {:.2f}'
     .format(clf.score(X_train, y_train)))
print('Accuracy of RF classifier on test set: {:.2f}'
     .format(clf.score(X_test, y_test)))
```

```
Accuracy of RF  classifier on training set: 0.98
Accuracy of RF classifier on test set: 0.80
```

In [79]:
```
from sklearn.metrics import f1_score,recall_score,precision_score
print(f1_score(y_test,pred,average='micro'))
print(recall_score(y_test,pred,average='micro'))
print(precision_score(y_test,pred,average='micro'))
```

```
0.8000000000000002
0.8
0.8
```

Considering all the three the model we find Logistic Regression is performing well as compare to rest two.
So, I choose **Logistic Regression** it may vary if you use larger dataset.

Setting Pipeline and saving

Now setting the pipeline and save it so we can directly access our model.

In [46]:
```
from sklearn.pipeline import Pipeline
from sklearn.externals import joblib
from sklearn import linear_model
import warnings
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [41]:
```
pipeline = Pipeline([
    ('vect', CountVectorizer()),
    ('tfidf', TfidfTransformer(norm='l2')),
    ('clf', linear_model.LogisticRegression(C=1e5)),
])
```

In [45]:
```
#saving the pipeline
filename = 'pipeline.sav'
joblib.dump(pipeline, filename)
```

Out[45]:
```
['pipeline.sav']
```

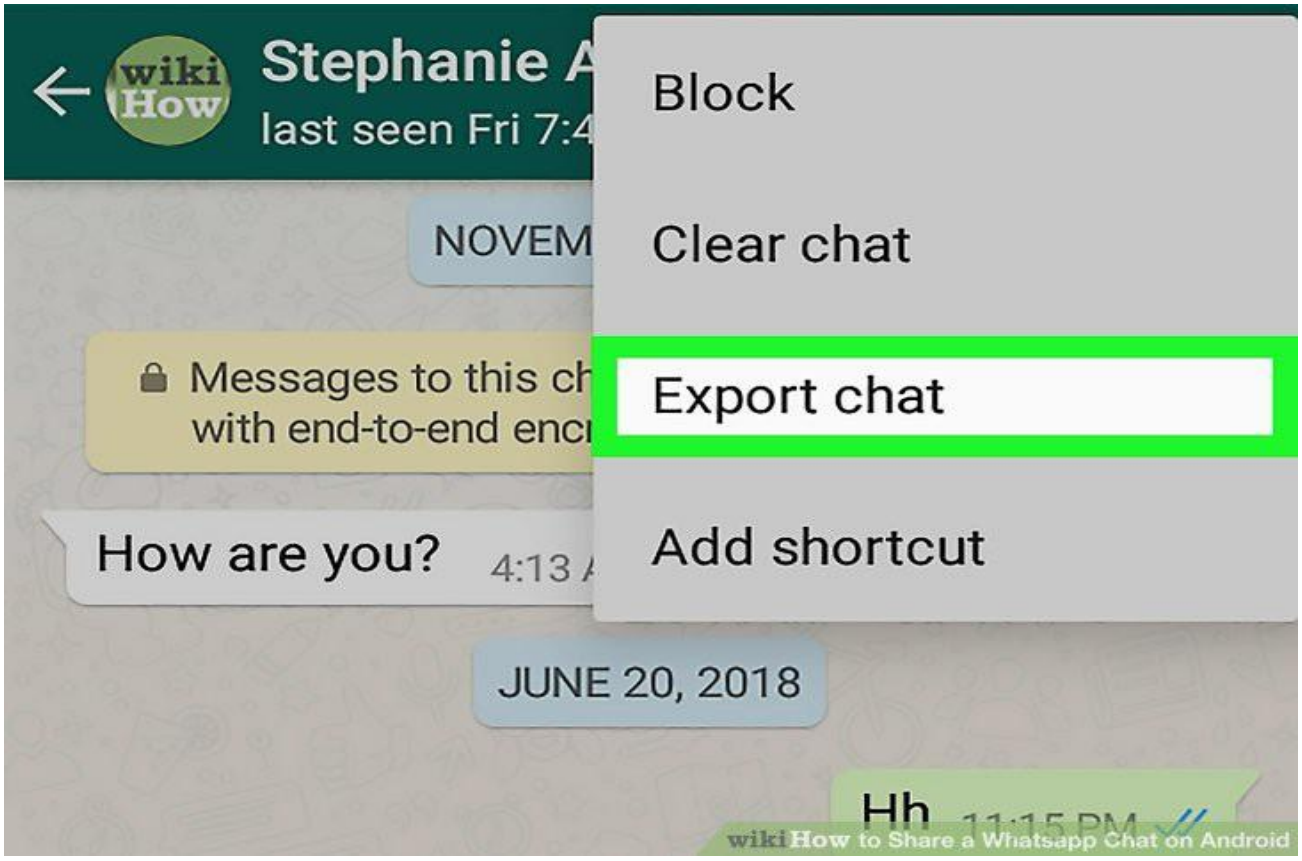
I had saved it as pipeline.sav

Export WhatsApp chat

Our project is about exporting WhatsApp data and passing it through our model and store our prediction in MongoDB database.

So, let's begin.

First, we need to export the chart:



Download the exported chat from your email inbox. It should resemble the following:

```
5/28/20, 3:37 PM - Modabbir: 9458762@Define Big Data and explain the Vs of Big Data
5/28/20, 3:38 PM - Modabbir: 9875201789@How is Hadoop related to Big Data
5/28/20, 3:39 PM - Modabbir: 7894561237@What is the difference between supervised and unsupervised machine learning?
5/28/20, 3:39 PM - Modabbir: 3215469870@How is KNN different from k-means clustering?
5/28/20, 3:40 PM - Modabbir: 9874566230@Define and describe the term FSCK.
5/28/20, 3:40 PM - Modabbir: 8794561233@Explain how a ROC curve works.
5/28/20, 3:41 PM - Modabbir: 7894561233@How React works? How Virtual-DOM works in React?
5/28/20, 3:41 PM - Modabbir: 9869741239@What is JSX?
```

Save the above file as file.txt

Building API and Storing prediction in MongoDB

Before moving further let's take a look on MongoDB. For storing your data there, you first have an account in Mongo Cloud. If you are using it for the first time kindly go through the article for setup.

https://medium.com/analytics-vidhya/how-to-upload-a-pandas-dataframe-to-mongodb-ffa18c0953c1

So, now you have successfully setup your account there and create a cluster. Make the database there giving name 'Projects' and give the name of the collection 'python_test'.

First import all the packages:

```
import pandas as pd
import numpy as np
from pymongo import MongoClient
import csv
import pandas as pd
import urllib
import os
import pymongo
import joblib
```

Here we had imported pymongo for creating a connection between python and mobgoDB, and joblib is for applying our saved model.
So, simply make the below function.
Let me explain the code:

- First, we pre-process it and store it in df.
- Then apply our saved model and store the prediction in result
- Convert result in to original labels
- Store it in MongoDB

```
def final(link,feature,filename,path,db_name,coll_name):
    df = pd.read_csv(link,header=None)
    df = df.drop(0,1)
    df[feature]=df[1].str.split('@').str.get(1)
    df=df.drop(1,1)
    df=df.dropna()

    #applying model
    loaded_model = joblib.load(filename)
    result = loaded_model.predict(df[feature])
    df['results']=result
    df['results'].replace({0:'ML', 1:'ReactJs', 2:'BigData'}, inplace=True)

    #storing result in mongoDB
    client = pymongo.MongoClient(path)
    db=client[db_name]
    collection = db[coll_name]
    df.reset_index(inplace=True)
    data_dict = df.to_dict("records")

    # Insert collection
    collection.insert_many(data_dict)
```

Congratulations!!! we are done.
Let's test it with your exported WhatsApp chat which you had stored in the txt file as file.txt

```
final('file.txt','questions','./pipeline.sav',\
      'mongodb+srv://modabbir24:onlyIIT#2018@cluster0-eergj.mongodb.net/test?retryWrites=true&w=majority',\
      'Projects','python_test')
```

Note:

- The path is different for every user so while putting its value follow the above-mentioned link for reference how to get the path.

Written by
Md Modabbir Tarique