A PROJECT REPORT ON

Fake News Classification

A project report submitted in fulfillment for the Diploma Degree in AI & ML

Under

Applied Roots with University of Hyderabad

Project submitted by

Surya Munjal

Under the guidance of mentor: **Neil Fowler**

University of Hyderabad

<u>Declaration of Authorship</u>

We hereby declare that this thesis titled" Fake News Classification" and the work presented by the undersigned candidate, as part of Diploma Degree in AI & ML.

All information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name: Surya Munjal
Thesis Title: Fake News Classification

CERTIFICATE OF RECOMMENDATION

We hereby recommend that the thesis entitled “Fake News Classification” prepared undermy supervision and guidance by Surya Munjal be accepted in fulfilment of the requirement for awarding the degree of Diploma in AI & ML Under applied roots with University of Hyderabad. The project, in our opinion, is worthy for its acceptance.

Mentor : Neil Flower

Under Applied roots with

University of Hyderabad

ACKNOWLEDGEMENT

I would like to acknowledge and appreciate the guidance provided by all the Faculties and mentors who have been key inspiration and motivational factor behind this project “Fake News Classification”. From sharing their valuable expertise and making the resources available have been an important factor behind the completion of this project.

Also, I would like to express my sincere thanks to my mentor: **Neil Flower** for assisting me and providing valuable insights and knowledge which helped me in completion of this project.

Name: Surya Munjal

PROBLEM DEFINITION

- In today's world of twitter, Facebook, watsapp and all other social media handles, people have become too prone to fake news. There are really concerning times ahead with widespread of hate speech and misleading news in traditional media and other social media handles. People often believe what they read. Fake news often tends to play with people's emotions and also have a real world impact.

  With ever increasing highly sensational and eye-catching headlines created by new channels aimed to attract masses; a fake news can stir an amplified unrequired reaction anytime.

- With the influx of huge amounts of information being generated every day at various fronts, manually detecting which news is fake is tedious task which will take huge amount of time. This motivated researchers to establish a model to automate the process and identify fake news patterns in news articles and media

- Our main purpose would be to come up with a solution that can be utilized by users to detect and filter out sites containing false and misleading information. Our main Objective is a Binary Classification task of classifying a news as fake or real.

- With the emergence of NLP and many other technologies, this problem has been studied over the past few years and people are coming up with new solutions to improve the effectiveness of the task in hand. From Standard Baseline Models like Naïve Bayes to complex State of the art deep learning architecture like BERT and LSTM are being used to improve the model key performance metric.

Dataset

The link for the dataset is
https://www.kaggle.com/datasets/akshayaki/fakenews

- dataset is 137 MB in size.
- There are 5 columns in our dataset
    - title: this is basically title of our new
    - text: text related to the news. Think of this as a subheading to title
    - Subject: This is like a category of news (politics, Middle-east. Etc)
    - Date: this is date on which news was broadcasted
    - Label: this is our output label (real or fake)

Things we know

- this is a Binary classification task
- Cost of misclassification is really high.
- we need probabilities of our class label i.e., what is the probability of a news being to the class fake or how much sure are we that the news is fake.
- We need some sort of interpretability

Business Metric

- **Log_loss** is our primary KPI. (prob score might be of importance)
- Confusion Matrix would be our secondary KPI. This would give us indication about our false positives, false negatives.
- We can also use accuracy score as our metric if our dataset is not imbalanced.
- precision and recall can be of importance here.

The code for implementation of accuracy for binary classification task from scratch is

```
losses = []
for yt,yp in zip(label,probs):
    first_part = yt * np.log(yp)
    second_part = (1-yt) * np.log(1-yp)
    merge = -1 * (first_part + second_part)
    losses.append(merge)
return np.mean(losses)#Taking the mean or average of the losses
```

REAL WORLD CHALLENGES AND CONSTRAINTS

1. LATENCY

The speed at which fake news spreads cannot be imagined. Hence it is very important to stop the fake news at very early stage. Thus, Latency can be thought of as a big constraint in our problem.

It's not like we want our fake news classifier output in 1ms or 10ms, but we want decent latency given a new news (query point).

2. INTERPRETEBILITY

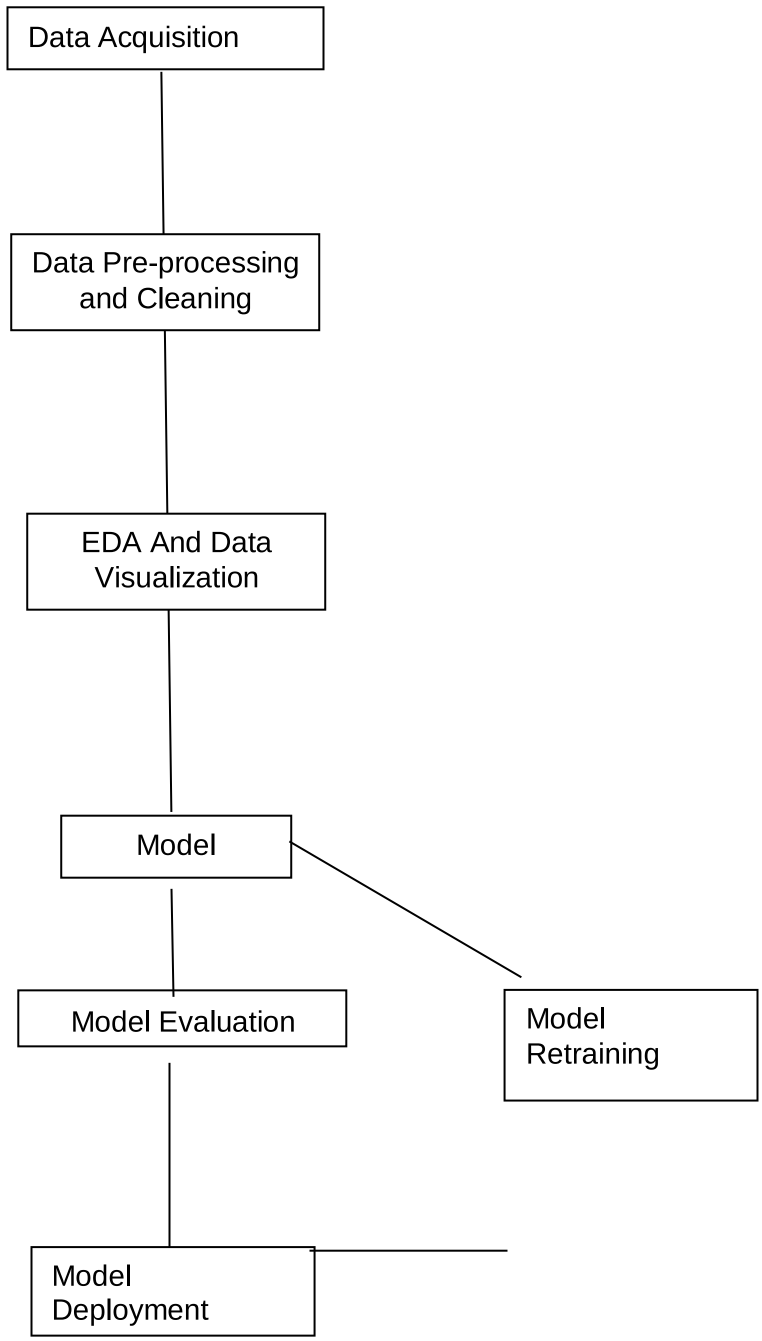
   We want our model to be interpretable. If a news is declared as fake, we want to know based on which features or words it is declared as fake.
   Some simple Machine learning are interpretable by themselves.

   Suppose if we take example of Logistic regression. The results of model can be easily interpretable by the trained weights.

   For more complex machine learning and deep learning models we can use techniques like LIME and SHAP to interpret our results.

3. To come up with new advanced features for machine learning task which can improve the performance our model.

Model Architecture



Data Acquisition: this step basically refers to acquiring data from whatever possible sources for our model.

Data Pre-processing: After acquiring the data, we need to pre- process our data.

Some Common steps in data pre-process include:

- Removing alphanumeric characters
- Removing html tags
- Stemming or lemmatization

EDA And Data visualization: this common steps in eda are

- Doing univariate and bivariate analysis of features.
- Checking for missing values
- Checking if our data set is imbalanced
- Checking how is our data distributed.

Model Training:

Throughout are project, we would analyzing distinct techniques that are being used previously for Fake News detection. Throughout the last decade, this problem of fake news classifier has been solved through many approaches. From simpler machine learning models like naïve bayes, logistic regression to complex STATE OF THE ART models like BERT, researchers have been trying to come up with new solutions to the problem. TF-IDF, Bag of Word (BOW) and Word2Vec have been used for basic NLP task. Nlp will help us in task of our feature extraction. We might create new features like length of news which might be helpful in improving our model performance.
This being a binary classification task, can be solved by several approaches. In this project, we would start with simpler Machine learning models like Naïve Bayes, SVM and Logistic regression. We know that naïve bayes works well on text classification data. So, we can consider Naïve bayes as our base Line model for evaluation.

Several machine learning approaches that we would be experimenting with are

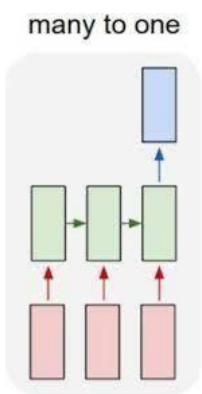
- Logistic regression
- Linear svm
- Naïve Bayes
- Decision Tree
- Random Forest
- XgBoost

After going through all the machine learning models, we might jump to using deep learning architectures which might help us in improving our metric score. Remember with advances of deep learning, we don’t need to explicitly create features for our task which a big plus in any problem statement

Deep learning approaches I might use include

- LSTM AND GRU
- BI-DIRECTIONAL RNN
- BERT (STATE OF THE ART)

We would be using architecture like many to one for LSTM AND RNN MODELS



We would have special focus on our model interpretability while we perform our classification task using different models.

<u>Model Evaluation:</u> This will be done based on business metric we defined earlier. For every Model we will calculate our performance and see which model is working best for our particular case.

<u>Model Retraining:</u>

After deploying models, the task of any data scientist is not over. We need to monitor our model performance. Model performance monitoring is the process of tracking the performance of your machine learning model based on live data in order to identify potential issues that might impact the business.

After monitoring and tracking your model performance, we can retrain our machine learning model. The objective is to ensure that the quality of your model in production is up to date.

-

<u>Exploratory Data Analysis</u>

How are data looks

```
[ ] df.head()
```

	Unnamed: 0	title	text	subject	date	Labels
0	0	GRAPHIC RIOT VIDEOS EXPOSE THUGS ATTACKING ELD...	youngers these days are becoming so moist pic....	politics	Sep 22, 2016	Fake
1	1	BIG BROTHER: FEDS WANT YOUR DOCTOR TO WARN YOU...	totally out of bounds! This is so wrong and so...	Government News	Jun 26, 2015	Fake
2	2	BLACK LIVES MATTER TERRORISTS May Be Shut Down...	ter ror ist ter r st/ noun a person who uses ...	politics	Dec 23, 2015	Fake
3	3	EMBARRASSING: Obama Explains How He Will 'Rebu...	re buke r byo ok/ verb 1. express sharp disapp...	left-news	Nov 25, 2015	Fake
4	4	OHIO ELECTOR TORCHES Anti-Trump Letters He Rec...	pic.twitter.com/KMnLrwB6t1 Richard K. Jones (...	politics	Dec 21, 2016	Fake

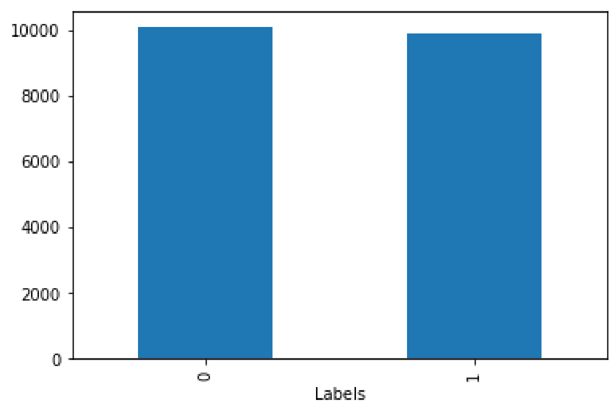
```
[ ] df.shape
(20000, 6)
```

Checking For Imbalanced Dataset

```
percentage of fake data points are 50.4 %
percentage of true data points are 49.6 %
```

```
df.groupby("Labels")['Labels'].count().plot.bar()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1947d0d310>
```



We can see the dataset is not imbalanced.

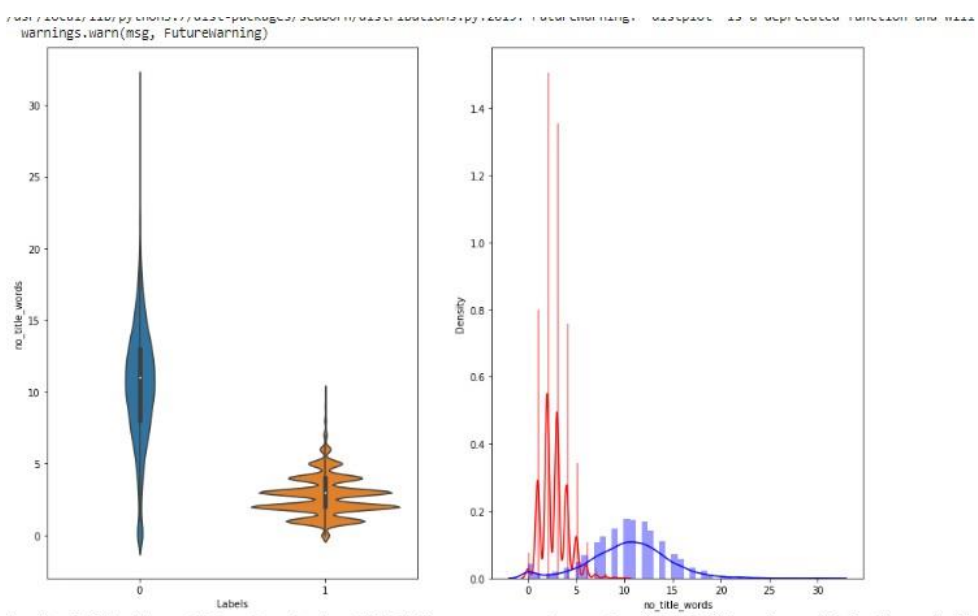
BASIC FEATURE EXTRACTION

We have constructed these 9 new features which would help us in improving our metric.

- title_words
- text_words
- title_length
- text_item
- count_punct
- no_title_words
- unique_words
- has_url
- no of special character

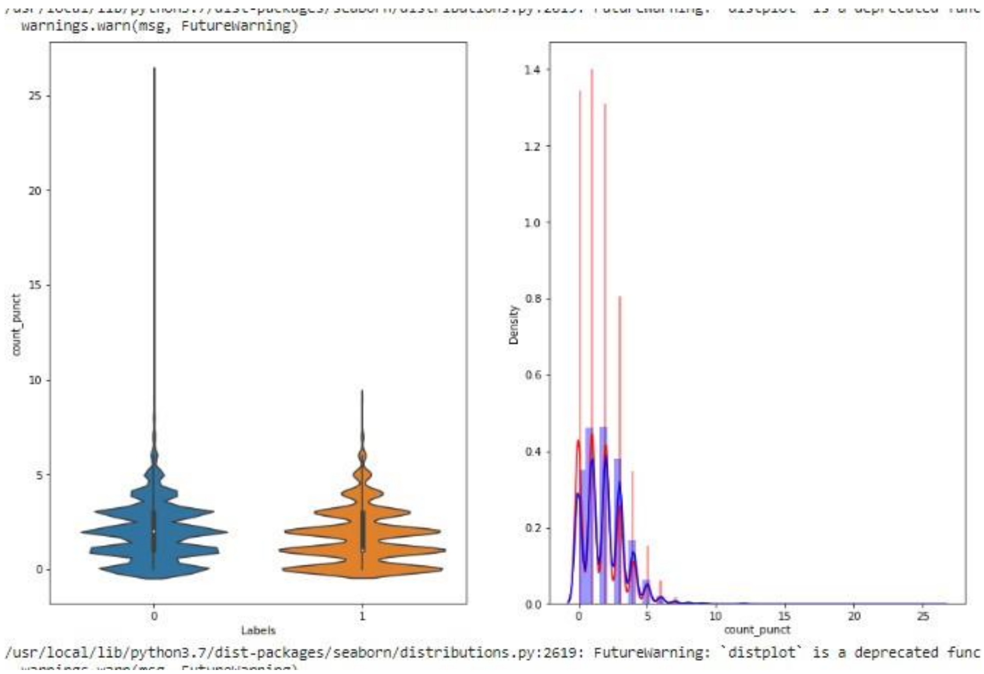
<u>plot of some of the created features:</u>

‘no_title_words’ with respect to our labels.

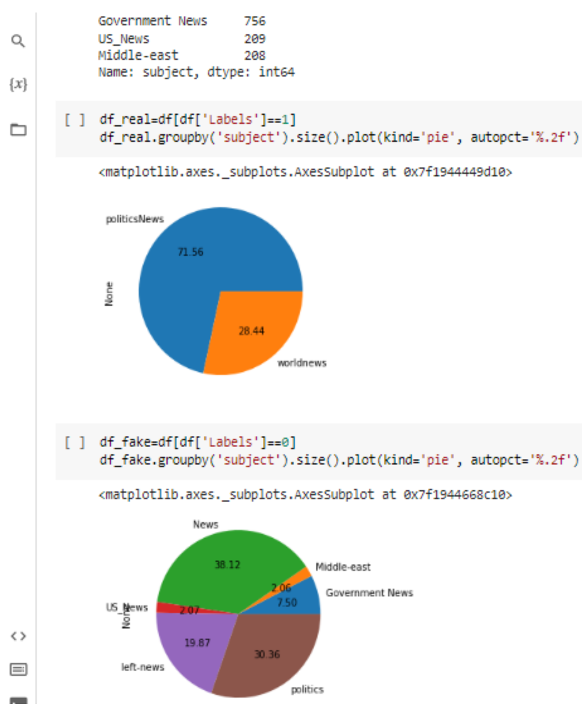


As we can se from plot, this feature can be useful for our model

Plot of count_punct with respect to our output label shows that this feature is not useful in classification task



Analyzing ‘Subject’ categorical column



This showed that how different subject values are present with respect to fake and real news

Plot of has_url feature

This shows that having a URL in news is a perfect indication for it to be classified as fake news.

<u>**Word Clouds**</u>

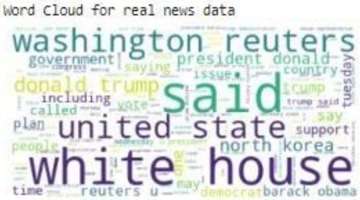
Fake News and real news show words are which are more common in which group.

```
plt.imshow(wc, interpolation='bilinear')
plt.axis("off")
plt.show()
```

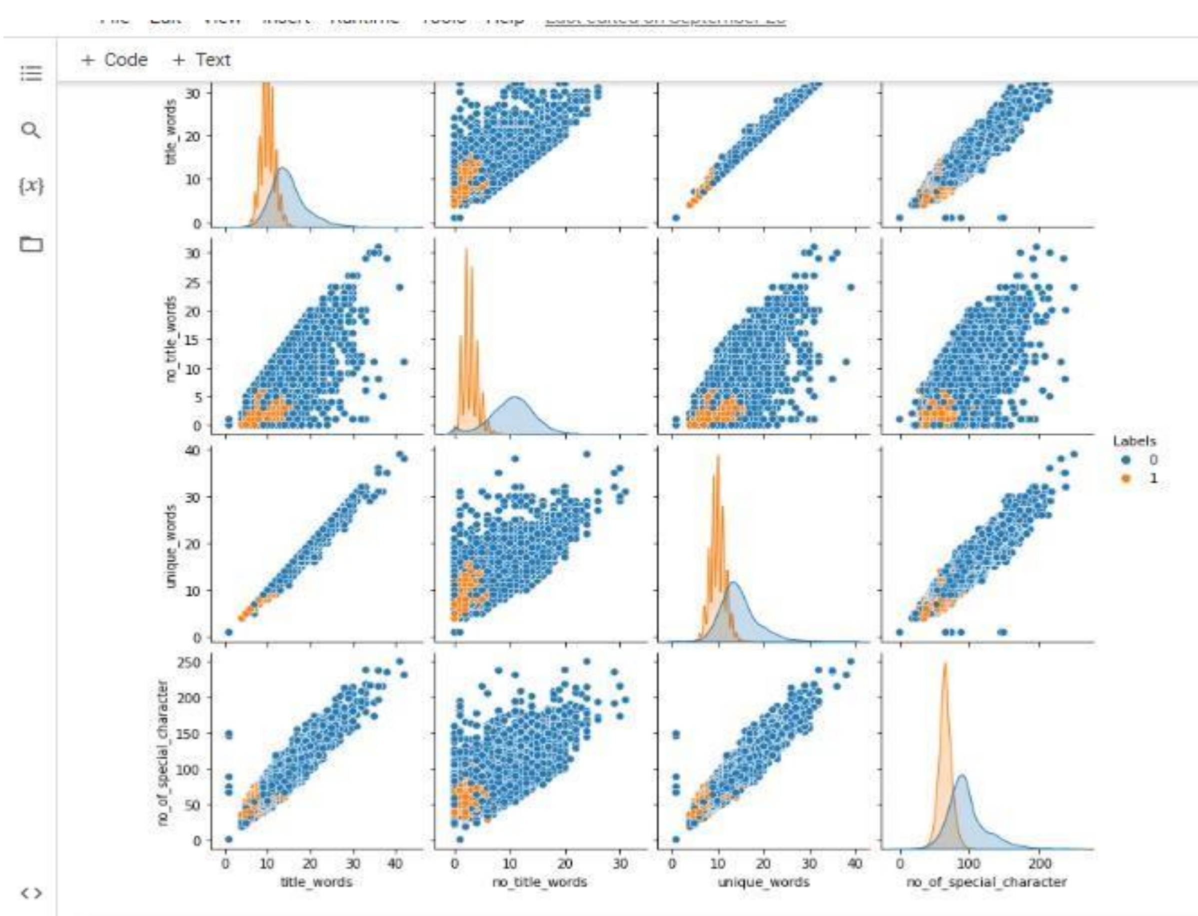
Word Cloud for Fake news


```
[ ] wc = WordCloud(background_color="white", max_words=len(textn_w))
    wc.generate(textn_w)
    print ("Word Cloud for real news data")
    plt.imshow(wc, interpolation='bilinear')
    plt.axis("off")
    plt.show()
```

Word Cloud for real news data

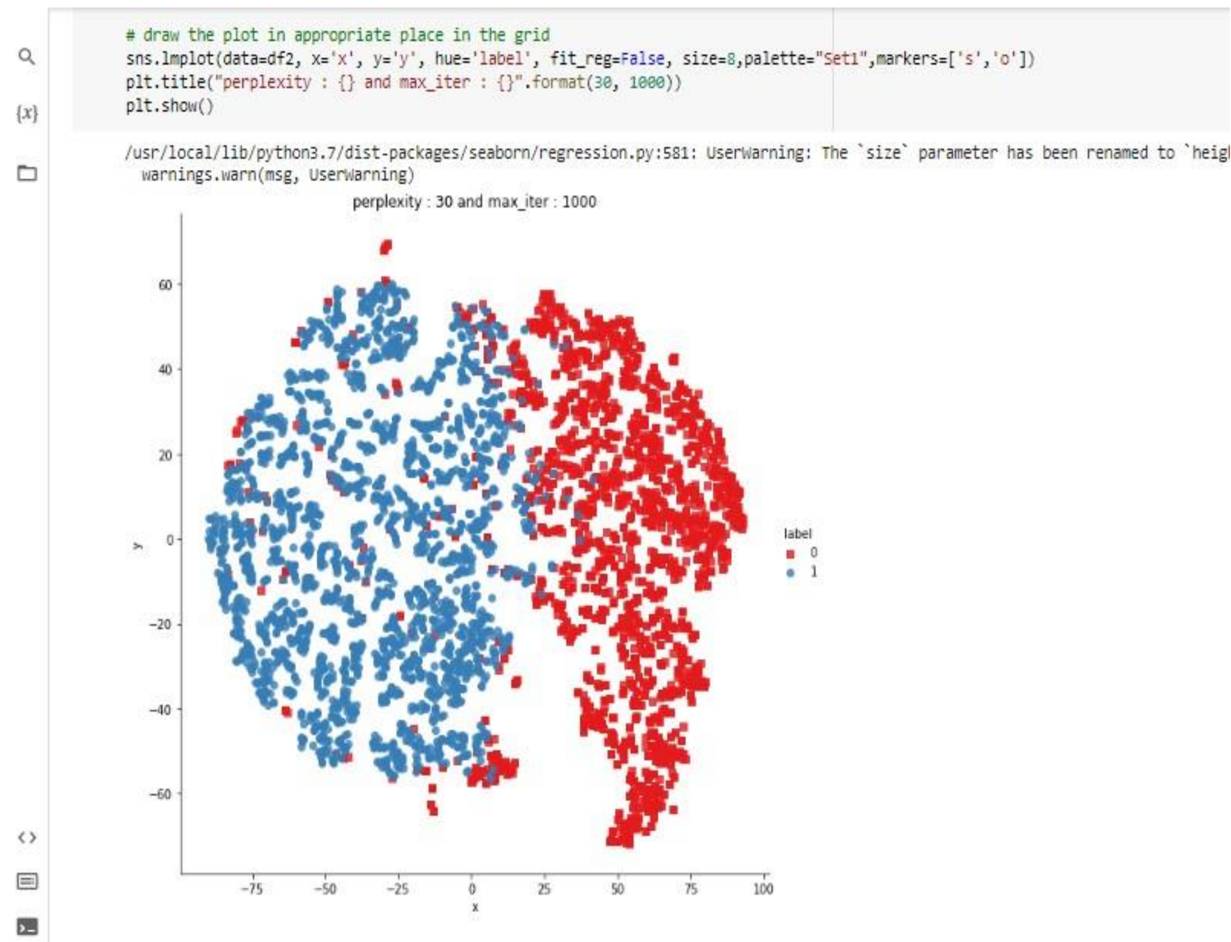


Pair Plots helps us visualizing our data using 2 features at a time.



Using TSNE for visualization

Running tsne shows that the features we have created are helpful in classification of our news as fake or real.



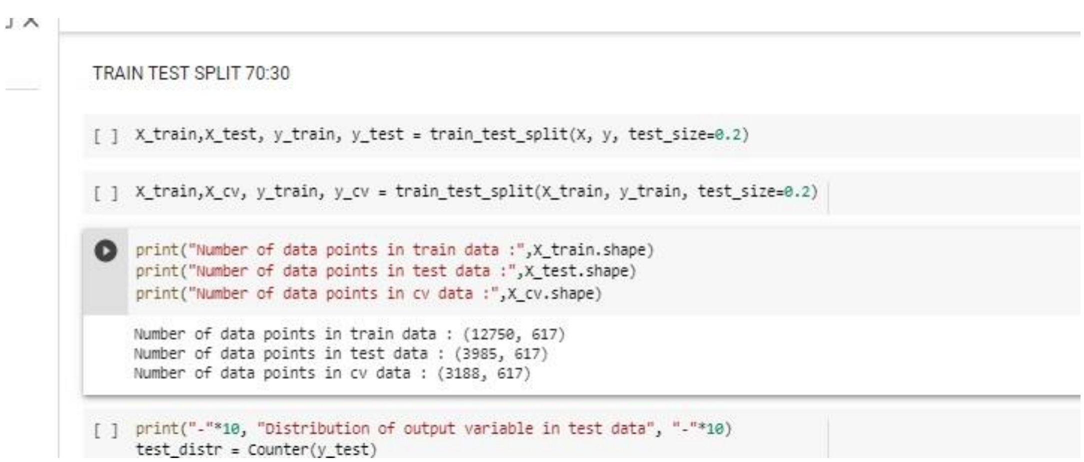
Data splitting

The model prediction can only be verified if we check if our results are accurate or not. This can be done by performing validation on the training data itself. So, the train data set must be split so that one part of it can be used to train the model while the other part is used for validation.

Here we used Simple Hold-Out Strategy (splitting data into Train and Test (with 70-30)

There are many methods for splitting the data and doing model validation like

- Simple Hold-Out Strategy (splitting data into Train and Test (with either 80-20 or 70-30 split)
- Three-way-hold-out strategy (splitting data into Training,Validation and Testing (50-20-30 split)
- Cross Validation (K-Fold cross-validation or RepeatedKfold etc)
- Leave-One-Out
- Grid Search etc.



TRAIN TEST SPLIT 70:30

```
[ ] X_train,X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

[ ] X_train,X_cv, y_train, y_cv = train_test_split(X_train, y_train, test_size=0.2)

print("Number of data points in train data :",X_train.shape)
print("Number of data points in test data :",X_test.shape)
print("Number of data points in cv data :",X_cv.shape)

Number of data points in train data : (12750, 617)
Number of data points in test data : (3985, 617)
Number of data points in cv data : (3188, 617)

[ ] print("-"*10, "Distribution of output variable in test data", "-"*10)
test_distr = Counter(y_test)
```

Step 8: Modelling

Classification Models

The major aim of this project is to predict which of the customers will have their loan paid or not. Therefore, this is a supervised classification problem to be trained with algorithms like:

The Problems are solved using below machine learning techniques:

1. Logistic Regression

2. Linear SVM

3. XGBOOST

The Problems are solved using below Deep Learning techniques:

- LSTM
- BIDERECTIONAL RNN
- BERT ENCODING

Logistic Regression

Logistic regression is a standard industry algorithm that is commonly used in practice because of its simplicity and balanced error distribution. It is a binary classification technique that generates one of two variables as its result, The logistic regression formula is shown below:

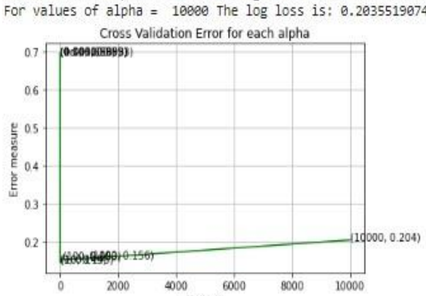
```
fig, ax = plt.subplots()
ax.plot(alpha, log_error_array,c='g')
for i, txt in enumerate(np.round(log_error_array,3)):
    ax.annotate((alpha[i],np.round(txt,3)), (alpha[i],log_error_array[i]))
plt.grid()
plt.title("Cross Validation Error for each alpha")
plt.xlabel("Alpha i's")
plt.ylabel("Error measure")
plt.show()


best_alpha = np.argmin(log_error_array)
clf = SGDClassifier(alpha=alpha[best_alpha], penalty='l2', loss='log', random_state=42)
clf.fit(X_train, y_train)
sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
sig_clf.fit(X_train, y_train)

predict_y = sig_clf.predict_proba(X_train)
print('For values of best alpha = ', alpha[best_alpha], "The train log loss is:",log_loss(y_train, predict_y, labels=clf.classes_, eps=1e-15))
predict_y = sig_clf.predict_proba(X_test)
print('For values of best alpha = ', alpha[best_alpha], "The test log loss is:",log_loss(y_test, predict_y, labels=clf.classes_, eps=1e-15))
predicted_y =np.argmax(predict_y,axis=1)
print("Total number of data points :", len(predicted_y))
plot_confusion_matrix(y_test, predicted_y)
```

```
For values of alpha =  1e-05 The log loss is: 0.6930316407365363
For values of alpha =  0.0001 The log loss is: 0.6930316407365363
For values of alpha =  0.001 The log loss is: 0.6930316407365363
For values of alpha =  0.01 The log loss is: 0.6930316407365363
For values of alpha =  0.1 The log loss is: 0.6930316407365363
For values of alpha =  1 The log loss is: 0.14459997218040904
For values of alpha =  10 The log loss is: 0.14471760125018787
For values of alpha =  100 The log loss is: 0.15313128306345927
For values of alpha =  1000 The log loss is: 0.15550376022353352
For values of alpha =  10000 The log loss is: 0.20355190744656484
```

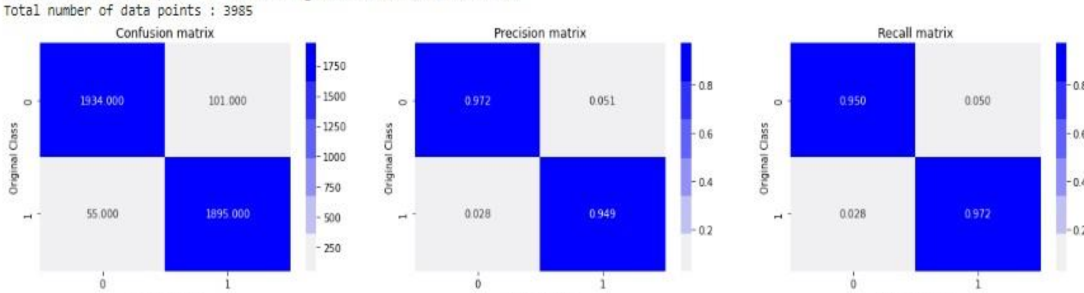
Help All changes saved

+ Code + Text

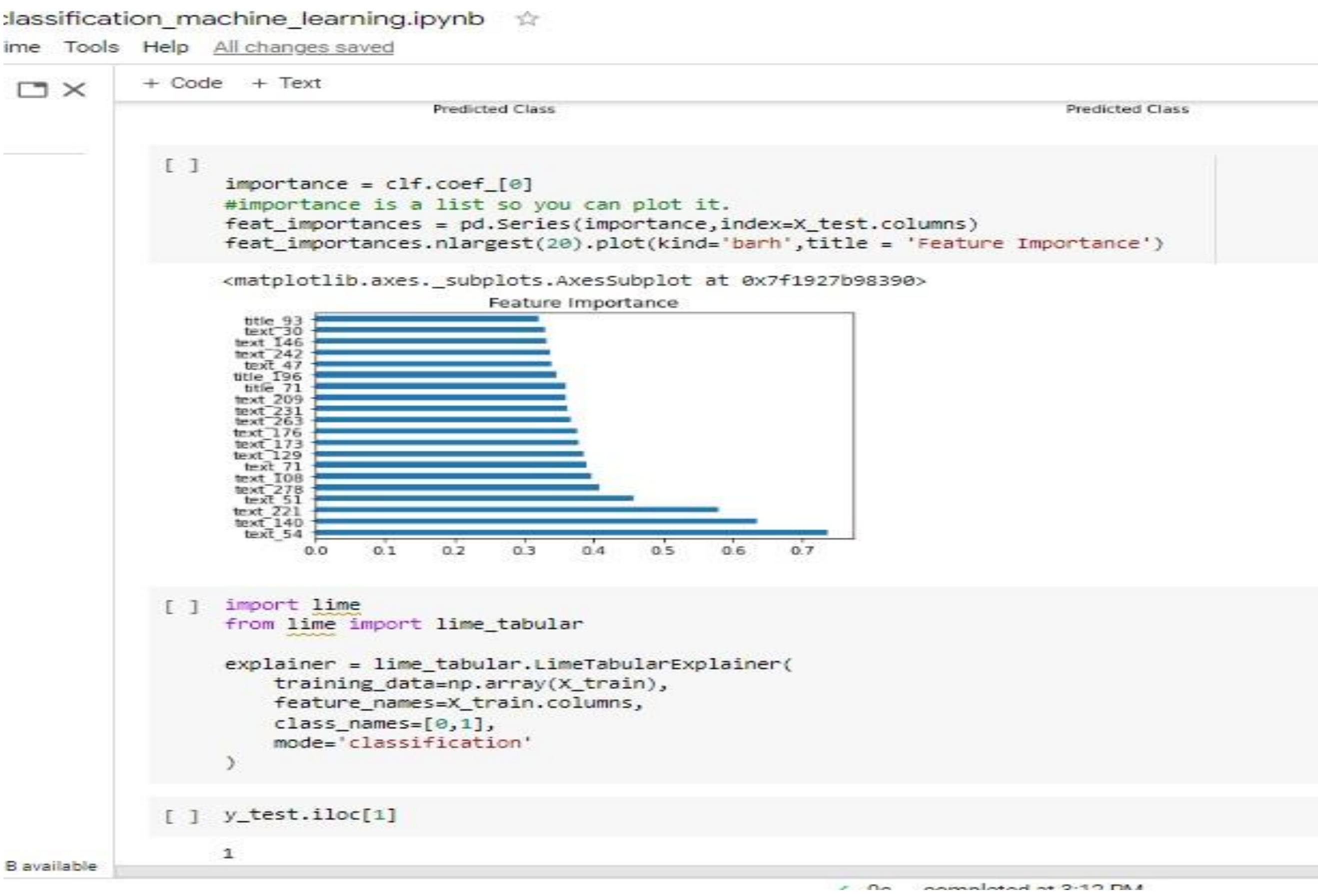
```
For values of alpha =  1000 The log loss is: 0.15550376022353352
For values of alpha =  10000 The log loss is: 0.20355190744656484
```



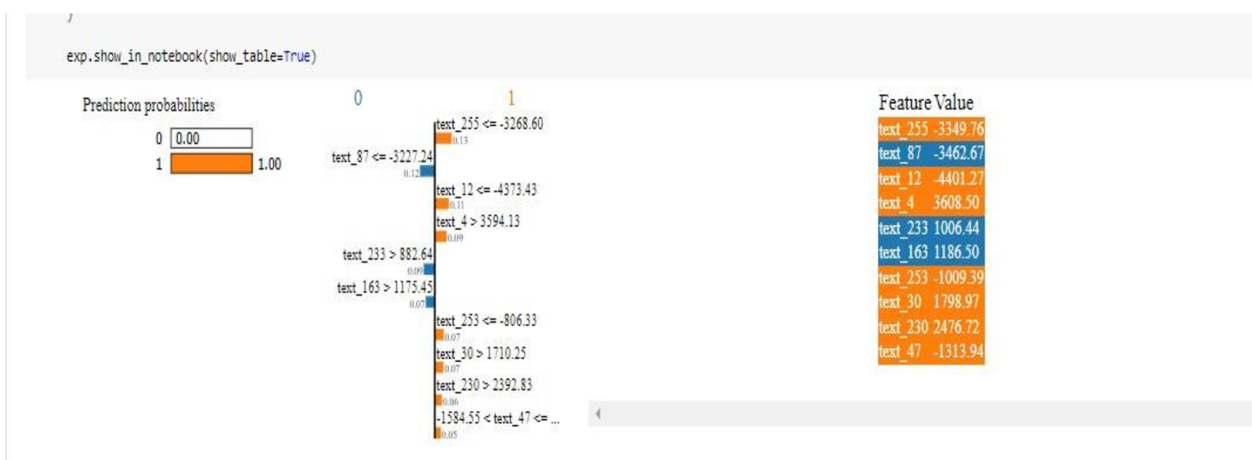

```
For values of best alpha =  1 The train log loss is: 0.11966093709370937
For values of best alpha =  1 The test log loss is: 0.1375043512153481
Total number of data points : 3985
```



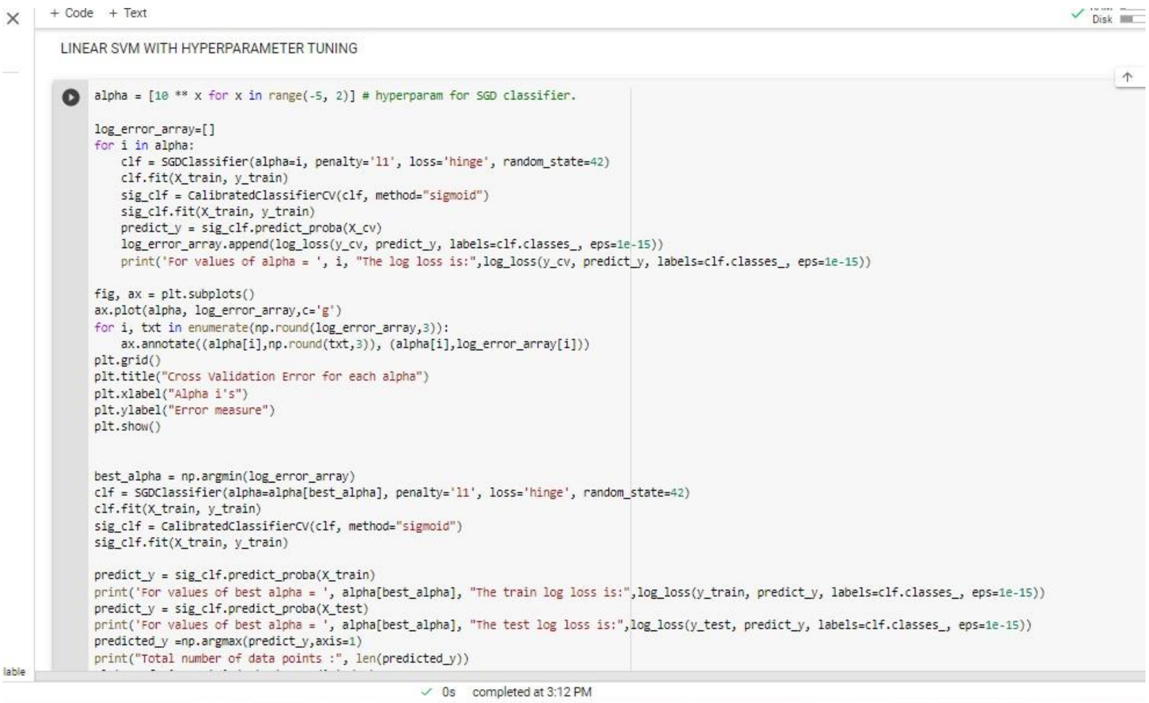
FEATURE IMPORTANCE IN LOGISTIC REGRESSION



Also using the lime module



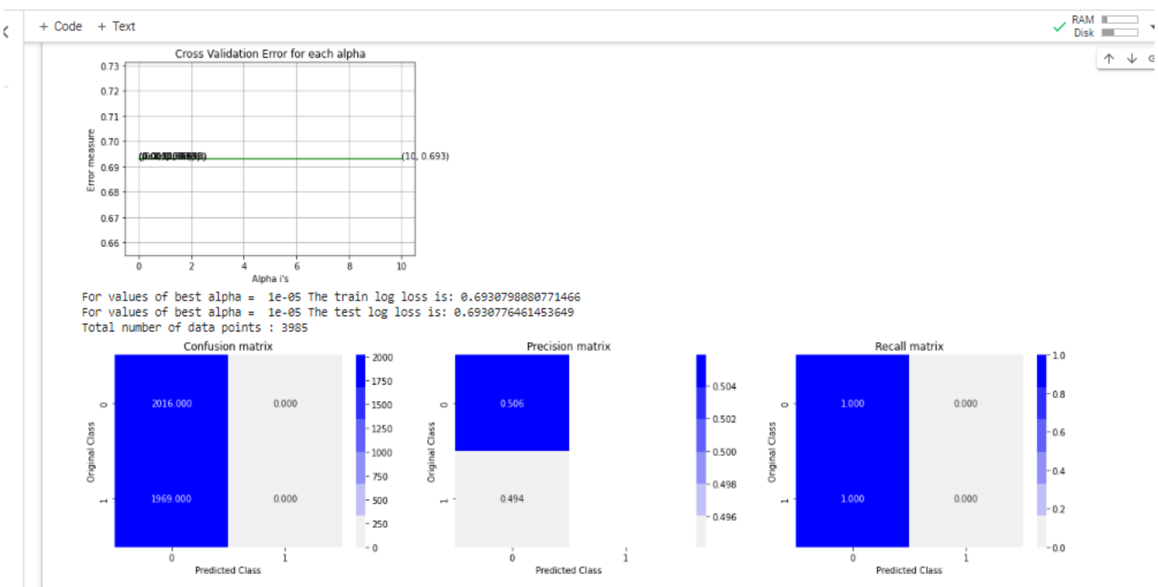
LINEAR SVM WITH HYPERPARAMETER TUNING


+ Code + Text

LINEAR SVM WITH HYPERPARAMETER TUNING

```
alpha = [10 ** x for x in range(-5, 2)] # hyperparam for SGD classifier.

log_error_array=[]
for i in alpha:
    clf = SGDClassifier(alpha=i, penalty='l1', loss='hinge', random_state=42)
    clf.fit(X_train, y_train)
    sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
    sig_clf.fit(X_train, y_train)
    predict_y = sig_clf.predict_proba(X_cv)
    log_error_array.append(log_loss(y_cv, predict_y, labels=clf.classes_, eps=1e-15))
    print('For values of alpha = ', i, "The log loss is:",log_loss(y_cv, predict_y, labels=clf.classes_, eps=1e-15))

fig, ax = plt.subplots()
ax.plot(alpha, log_error_array,c='g')
for i, txt in enumerate(np.round(log_error_array,3)):
    ax.annotate((alpha[i],np.round(txt,3)), (alpha[i],log_error_array[i]))
plt.grid()
plt.title("Cross Validation Error for each alpha")
plt.xlabel("Alpha i's")
plt.ylabel("Error measure")
plt.show()


best_alpha = np.argmin(log_error_array)
clf = SGDClassifier(alpha=alpha[best_alpha], penalty='l1', loss='hinge', random_state=42)
clf.fit(X_train, y_train)
sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
sig_clf.fit(X_train, y_train)

predict_y = sig_clf.predict_proba(X_train)
print('For values of best alpha = ', alpha[best_alpha], "The train log loss is:",log_loss(y_train, predict_y, labels=clf.classes_, eps=1e-15))
predict_y = sig_clf.predict_proba(X_test)
print('For values of best alpha = ', alpha[best_alpha], "The test log loss is:",log_loss(y_test, predict_y, labels=clf.classes_, eps=1e-15))
predicted_y =np.argmax(predict_y,axis=1)
print("Total number of data points :", len(predicted_y))
```

✓ 0s completed at 3:12 PM



XGBOOST

XGBOOST ...

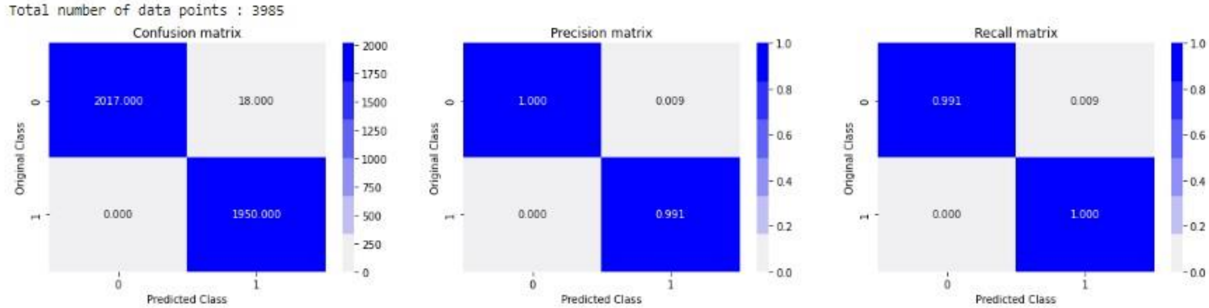
```
import xgboost as xgb
params = {}
params['objective'] = 'binary:logistic'
params['eval_metric'] = 'logloss'
params['eta'] = 0.02
params['max_depth'] = 4

d_train = xgb.DMatrix(X_train, label=y_train)
d_test = xgb.DMatrix(X_test, label=y_test)

watchlist = [(d_train, 'train'), (d_test, 'valid')]

bst = xgb.train(params, d_train, 100, watchlist, early_stopping_rounds=10, verbose_eval=5)

xgdmat = xgb.DMatrix(X_train,y_train)
predict_y = bst.predict(d_test)
print("The test log loss is:",log_loss(y_test, predict_y, labels=clf.classes_, eps=1e-15))
```

```
[0]	train-logloss:0.673969	valid-logloss:0.674151
Multiple eval metrics have been passed: 'valid-logloss' will be used for early stopping.

Will train until valid-logloss hasn't improved in 10 rounds.
[5]	train-logloss:0.588688	valid-logloss:0.589571
[10]	train-logloss:0.517616	valid-logloss:0.519226
[15]	train-logloss:0.457709	valid-logloss:0.459925
[20]	train-logloss:0.406662	valid-logloss:0.409463
[25]	train-logloss:0.362807	valid-logloss:0.366129
[30]	train-logloss:0.324723	valid-logloss:0.328606
[35]	train-logloss:0.291431	valid-logloss:0.295618
[40]	train-logloss:0.262089	valid-logloss:0.26648
[45]	train-logloss:0.236376	valid-logloss:0.240872
[50]	train-logloss:0.213324	valid-logloss:0.217991
[55]	train-logloss:0.192446	valid-logloss:0.197393
[60]	train-logloss:0.174077	valid-logloss:0.1792
[65]	train-logloss:0.157474	valid-logloss:0.162563
[70]	train-logloss:0.141288	valid-logloss:0.145811
```

.

```
[ ] predicted_y =np.array(predict_y>0.5,dtype=int)
    print("Total number of data points :", len(predicted_y))
    plot_confusion_matrix(y_test, predicted_y)
```

```
Total number of data points : 3985
```



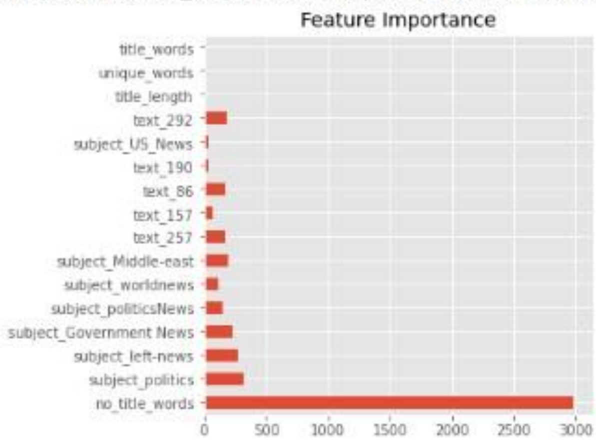

```
[ ] bst.get_score(importance_type='gain')
```

Xgboost interpretation

```
 'unique_words': 1.941070555}
```

```
[ ] importance=bst.get_score(importance_type='gain')
    #importance is a dict so you can plot it.
    feat_importances = pd.Series(importance.values(),index=importance.keys())
    feat_importances.plot(kind='barh',title = 'Feature Importance')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f19431c7090>
```




```
[ ]
```

Implementation using deep learning

LSTM

Tools Help All changes saved

+ Code + Text

```
37  2861  2704   516    29   151   107    76  1677  5431   438   370]
```

CREATING A LSTM MODEL

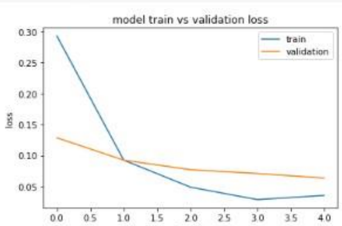
```
# create the model
embedding_vecor_length = 32
model = Sequential()
model.add(Embedding(total, embedding_vecor_length, input_length=max_length))
model.add(LSTM(8))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
print(model.summary())
#Refer: https://datascience.stackexchange.com/questions/10615/number-of-parameters-in-an-lstm-model

#log loss is  loss='binary_crossentropy' specifies that your model should optimize the log loss for binary classification.
#metrics=['accuracy'] specifies that accuracy should be printed out, but log loss is also printed out by default.
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 600, 32)           3361120

 lstm (LSTM)                 (None, 8)                 1312

 dense (Dense)               (None, 1)                 9

=================================================================
Total params: 3,362,441
Trainable params: 3,362,441
Non-trainable params: 0
_________________________________________________________________
None
```

```
[31] history=model.fit(X_train, y_train, epochs=5, batch_size=64,validation_split=0.3)
     # Final evaluation of the model
     scores = model.evaluate(X_test, y_test, verbose=0)
     print("Accuracy: %.2f%%" % (scores[1]*100))
```

+ Code + Text

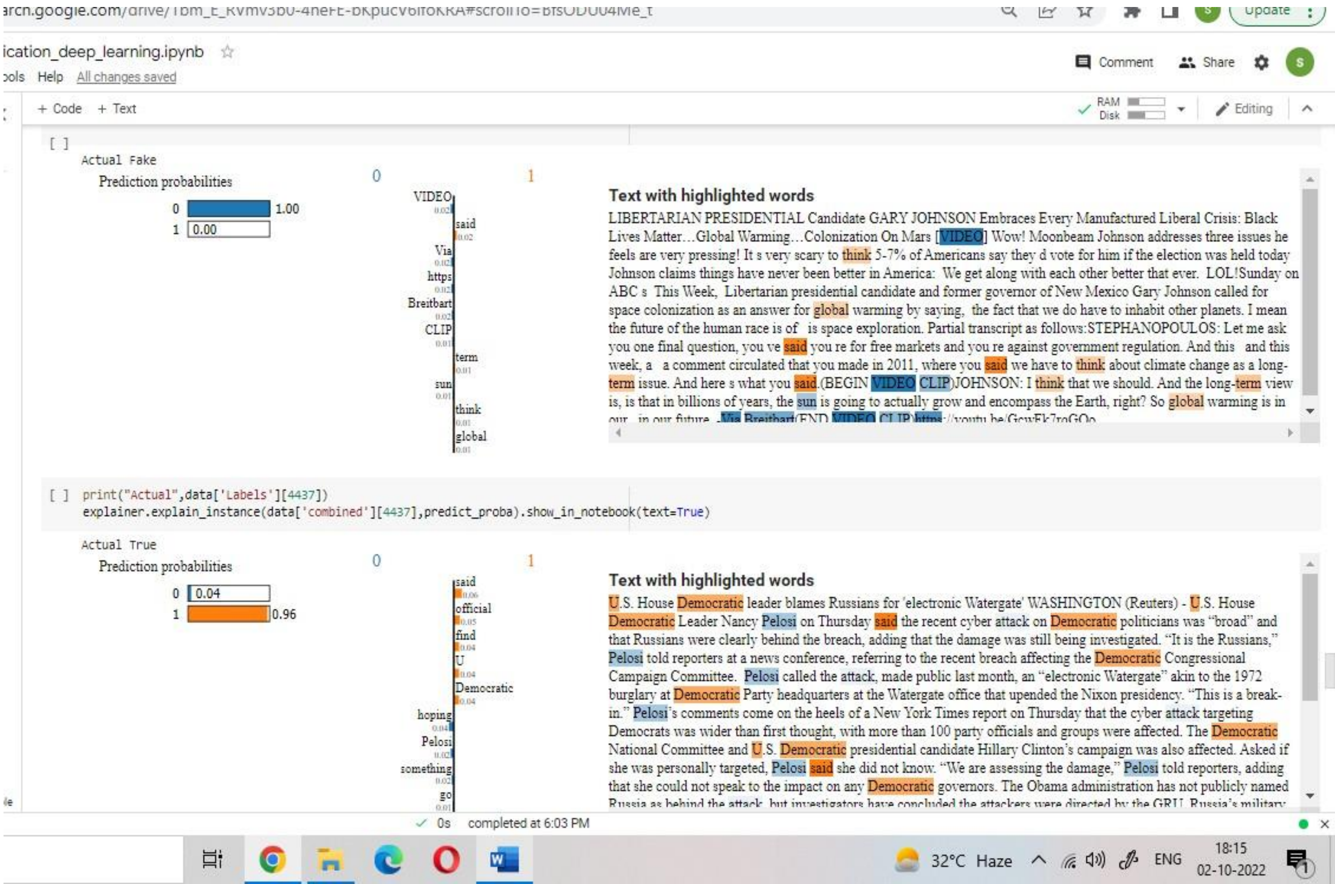
```
[39] # plot train and validation loss
     from matplotlib import pyplot
     pyplot.plot(history.history['loss'])
     pyplot.plot(history.history['val_loss'])
     pyplot.title('model train vs validation loss')
     pyplot.ylabel('loss')
     pyplot.xlabel('epoch')
     pyplot.legend(['train', 'validation'], loc='upper right')
     pyplot.show()
```




```
[47] from sklearn.metrics import confusion_matrix
     y_pred = model.predict(X_test)
     confusion_matrix = confusion_matrix(y_test, np.rint(y_pred))
```

```
[48] print(confusion_matrix)

     [[6973  121]
      [  80 6296]]
```

Using lime module to interpret results



Bidirectional RNN

LETS CREATE A BIDIRECTIONAL RNN MODEL

```
## Creating model
from tensorflow.keras.layers import Bidirectional
from tensorflow.keras.layers import Dropout
# create the model

embedding_vecor_length = 32
model1 = Sequential()
model1.add(Embedding(total, embedding_vecor_length, input_length=max_length))
model1.add(Bidirectional(LSTM(8)))
model1.add(Dropout(0.3))
model1.add(Dense(1, activation='sigmoid'))
model1.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
print(model1.summary())
```

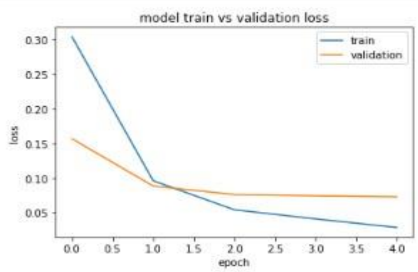
```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_1 (Embedding)     (None, 600, 32)           3361120

 bidirectional (Bidirectiona  (None, 16)               2624
 l)

 dropout (Dropout)           (None, 16)                0

 dense_1 (Dense)             (None, 1)                 17

=================================================================
Total params: 3,363,761
Trainable params: 3,363,761
Non-trainable params: 0
_________________________________________________________________
None
```

0s completed at 6:03 PM

Tools Help Save failed

+ Code + Text

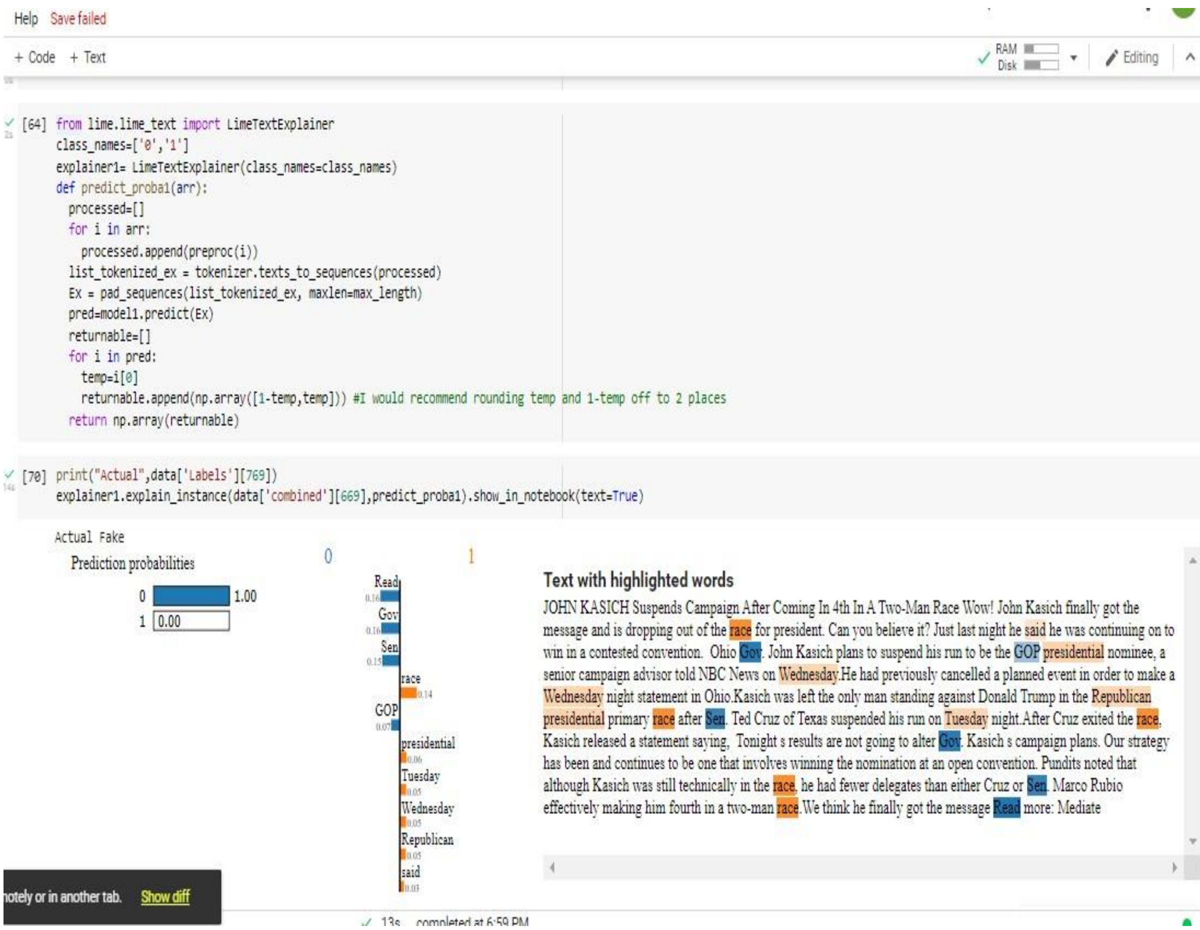
```
[56] pyplot.show()
```




```
from sklearn.metrics import confusion_matrix
y_pred1 = model1.predict(X_test)
confusion_matrix1 = confusion_matrix(y_test, np.rint(y_pred1))
print(confusion_matrix1)
from sklearn.metrics import classification_report
target_names = ['fake 0', 'real 1']
print(classification_report(y_test, np.rint(y_pred1),target_names=target_names))
```

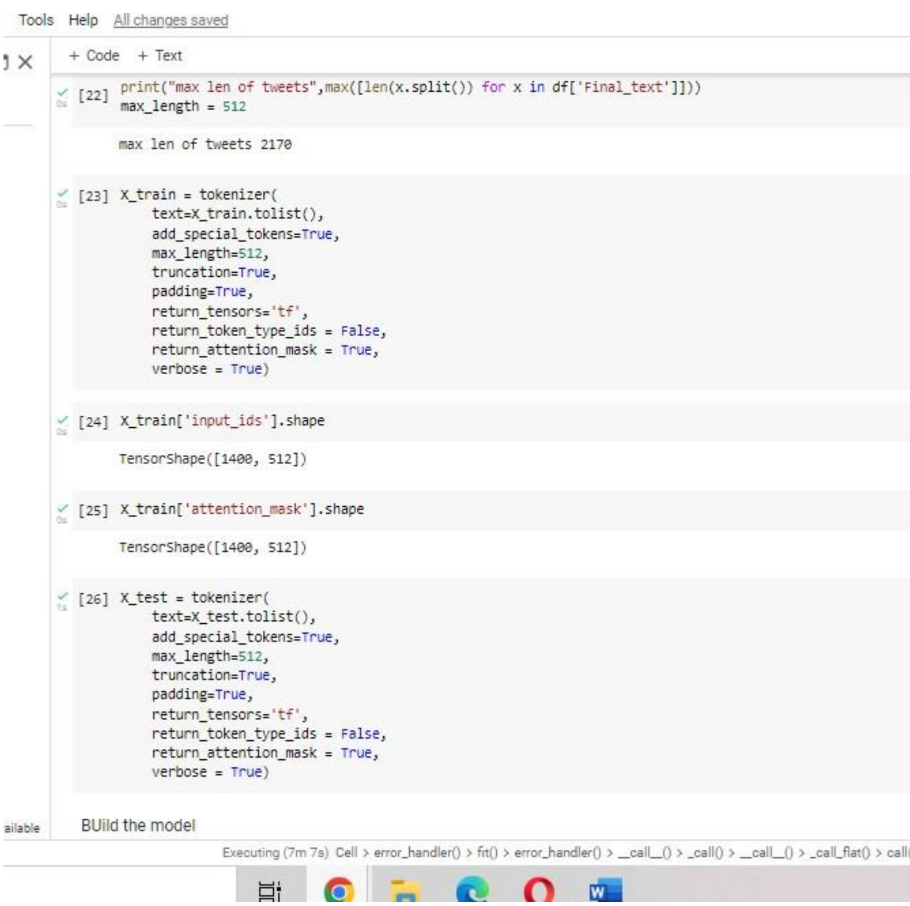
```
[[6962  132]
 [ 123 6253]]
              precision    recall  f1-score   support

      fake 0       0.98      0.98      0.98      7094
      real 1       0.98      0.98      0.98      6376

    accuracy                           0.98     13470
   macro avg       0.98      0.98      0.98     13470
weighted avg       0.98      0.98      0.98     13470
```

Interpreting the results through lime



BERT

Prepare the dataset


Tools Help All changes saved

+ Code + Text

[22]
```
print("max len of tweets",max([len(x.split()) for x in df['Final_text']]))
max_length = 512
```

```
max len of tweets 2170
```

[23]
```
X_train = tokenizer(
    text=X_train.tolist(),
    add_special_tokens=True,
    max_length=512,
    truncation=True,
    padding=True,
    return_tensors='tf',
    return_token_type_ids = False,
    return_attention_mask = True,
    verbose = True)
```

[24]
```
X_train['input_ids'].shape
```

```
TensorShape([1400, 512])
```

[25]
```
X_train['attention_mask'].shape
```

```
TensorShape([1400, 512])
```

[26]
```
X_test = tokenizer(
    text=X_test.tolist(),
    add_special_tokens=True,
    max_length=512,
    truncation=True,
    padding=True,
    return_tensors='tf',
    return_token_type_ids = False,
    return_attention_mask = True,
    verbose = True)
```

BUild the model

Executing (7m 7s) Cell > error_handler() > fit() > error_handler() > __call__() > _call() > __call__() > _call_flat() > call(

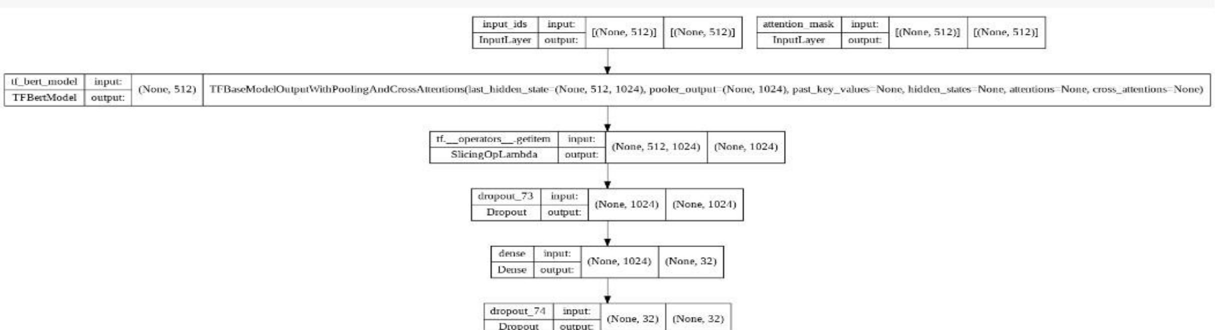
Build the model

Using BERT encoding and building a MLP with 1 hidden layer of 32 neuron
on top of it.

```
[27]

[28] max_len = 512
     import tensorflow as tf
     from tensorflow.keras.layers import Input, Dense

     input_ids = Input(shape=(max_len,), dtype=tf.int32, name="input_ids")
     input_mask = Input(shape=(max_len,), dtype=tf.int32, name="attention_mask")
     # embeddings = dbert_model(input_ids,attention_mask = input_mask)[0]

     last_hidden_state = bert(input_ids,attention_mask = input_mask)[0]

     cls_token=last_hidden_state[:, 0, :]
                                              #(0 is the last hidden states,1 means output)
     # out = tf.keras.layers.GlobalMaxPool1D()(embeddings)
     out = tf.keras.layers.Dropout(0.1)(cls_token)

     out = Dense(32, activation='relu')(out)
     out = tf.keras.layers.Dropout(0.1)(out)
     y = Dense(1,activation = 'sigmoid')(out)

     model = tf.keras.Model(inputs=[input_ids, input_mask], outputs=y)
     model.layers[2].trainable = False

[29] model.summary()

     Model: "model"
     ______________________________________________________________________________________
      Layer (type)                Output Shape          Param #    Connected to
     ======================================================================================
      input_ids (InputLayer)      [(None, 512)]         0          []

      attention_mask (InputLayer) [(None, 512)]         0          []

      tf_bert_model (TFBertModel) TFBaseModelOutputWi   335141888  ['input_ids[0][0]',
                                  thPoolingAndCrossAt               'attention_mask[0][0]']
                                  tentions(last_hidde
                                  n_state=(None, 512,
                                   1024),
```

Executing (8m 53s) Cell > error_handler() > fit() > error_handler() > __call__() > _call() > __call__() > _call_flat() > call() > quick_execute()

```
[30]     clipnorm=1.0)
         ...

     # Set loss and metrics
     loss = BinaryCrossentropy(from_logits = True)
     metric = BinaryAccuracy('accuracy'),
     # Compile the model
     model.compile(
         optimizer = 'adam',
         loss = loss,
         metrics = metric)

[31] plot_model(model, show_shapes = True)
```



Executing (11m 52s) Cell > error_handler() > fit() > error_handler() > __call__() > _call() > __call__() > _call_flat() > call() > quick_execute()

Conclusion:

We have successfully implemented fake news classification using machine Learning and deep learning algos. As we saw LSTM gave us the best log loss of 0.053 for test data.

We would be using lstm model for deployment purpose.

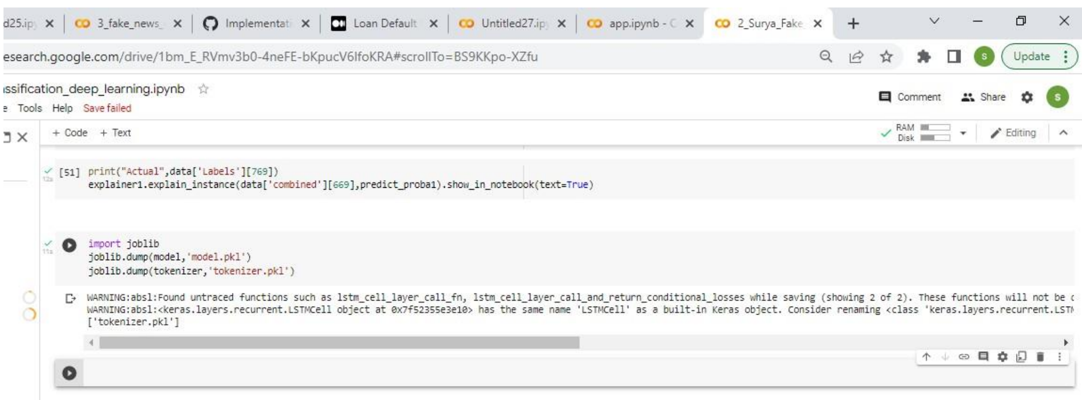
References

- https://www.sciencedirect.com/science/article/pii/S1877050918318210
- https://www.researchgate.net/publication/358015768_Fake_News_Classification_Based_on_Content_Level_Features
- https://www.kaggle.com/code/ruchi798/how-do-you-recognize-fake-news/notebook
- https://iopscience.iop.org/article/10.1088/1757-899X/1099/1/012040/pdf
- https://towardsdatascience.com/fake-news-classifier-e061b339ad6c
- https://www.ijert.org/survey-on-fake-news-detection-using-machine-learning-algorithms
- https://www.sciencedirect.com/science/article/pii/S2666307422000092
- https://www.edureka.co/community/171323/compute-log-loss-for-logistic-regression-from-scratch
- https://www.analyticsvidhya.com/blog/2022/03/fake-news-classification-using-deep-learning/
- https://www.analyticsvidhya.com/blog/2022/03/fake-news-classification-using-deep-learning/

1. Deployment and Productionization

The final phase of this project.is Deployment, here we are deploying the whole machine learning pipeline into a production system, into a real-time scenario.

In this final stage we need to deploy this machine learning pipeline to put of research available to end users for use. The model will be deployed in real world scenario where it takes continuous raw input from real world and predict the output.
We would be deploying our LSTM model because it gave us best results.



We have deployed using ngrok module which help us deploying our app.py file while working with colab.

+ Code + Text

```
[ ]  # import Flask from flask module
     from flask import Flask

     # import run_with_ngrok from flask_ngrok to run the app using ngrok
     from flask_ngrok import run_with_ngrok
     from flask import Flask, render_template , request


     app = Flask(__name__) #app name
     run_with_ngrok(app)
     '''
     @app.route("/")
     def hello():
         return "Hello Friends! from Pykit.org. Thank you! for reading this article."
     '''
     @app.route("/")
     def index():
         return render_template("index.html")


     @app.route('/predict', methods=['POST'])
     def predict():

       to_predict = request.form.to_dict()
       #to_predict_text = request.form.to_dict()
       news_title = preproc(to_predict['news_title'])
       news_text = preproc(to_predict['news_text'])

       clf = joblib.load('model_news.pkl')
       tokenizer = joblib.load('tokenizer.pkl')
       combined=news_title + news_text
       processed=[]
       processed.append(preproc(combined))
       list_tokenized_ex = tokenizer.texts_to_sequences(processed)
```

.

Help Last edited on September 7

+ Code + Text

```
       news_text = preproc(to_predict['news_text'])
[ ]
       clf = joblib.load('model_news.pkl')
       tokenizer = joblib.load('tokenizer.pkl')
       combined=news_title + news_text
       processed=[]
       processed.append(preproc(combined))
       list_tokenized_ex = tokenizer.texts_to_sequences(processed)
       Ex = pad_sequences(list_tokenized_ex, maxlen=600)
       pred = clf.predict(Ex)
       if pred[0]>=0.5:
           prediction = "Positive"
       else:
           prediction = "Negative"

       a = "prediction is" + " " + prediction

       return  a


     if __name__ == "__main__":
         app.run()

 * Serving Flask app "__main__" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
INFO:werkzeug: * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
 * Running on http://507d-34-125-69-119.ngrok.io
 * Traffic stats available on http://127.0.0.1:4040
INFO:werkzeug:127.0.0.1 - - [07/Sep/2022 13:29:00] "GET / HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [07/Sep/2022 13:29:00] "GET /favicon.ico HTTP/1.1" 404 -
INFO:werkzeug:127.0.0.1 - - [07/Sep/2022 13:29:26] "POST /predict HTTP/1.1" 200 -

[ ]
```

Not secure | f4b2-34-80-38-65.ngrok.io

fake news classifier : Sentiment Analysis

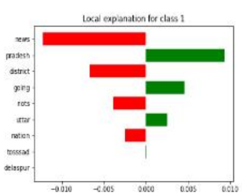
News Title

News Text

Submit

Fake News

WHY IT SHOWED THIS RESULT



Proj74ThesisRep....docx

predict.html

index.html

app (1).ipynb

app.ipynb