FAKE NEWS DETECTION USING NLP

TEAM MEMBER

Phase - 4 submission document

Project title: Fake news detection using nlp

Phase 4: Development part 2

Topic: Continue development and Evaluation for fake news prediction

FAKE NEWS PREDICTION USING NLP

Introduction:

Fake news detection is a hot topic in the field of natural language processing. In this article, we are using this dataset for news classification using NLP techniques.

We are given two input files. One with real news and the other one with fake news. Let us dig further into the given data. Our job is to create a model which predicts whether a given news is real or fake.

As this is a supervised learning problem, we are creating a target column named 'label' in both real and fake news data and concatenating them.

Natural Language Processing is among the hottest topic in the field of data science. Companies are putting tons of money into research in this field. Everyone is trying to understand Natural Language Processing and its applications to make a career around it.

Process:

Most text and documents contain many terms that are redundant for text classification,

such as stop words, misspellings, slangs, and so on.

Hence, data pre-processing has to be done before the data is sent to the classification models.

After that, the dataset's dimensionality is decreased in order to save time and storage space. When the dimensions are reduced, it becomes easier to visualise.

The data is then used to train classification models, which can be used to predict whether or not the presented data is fraudulent.
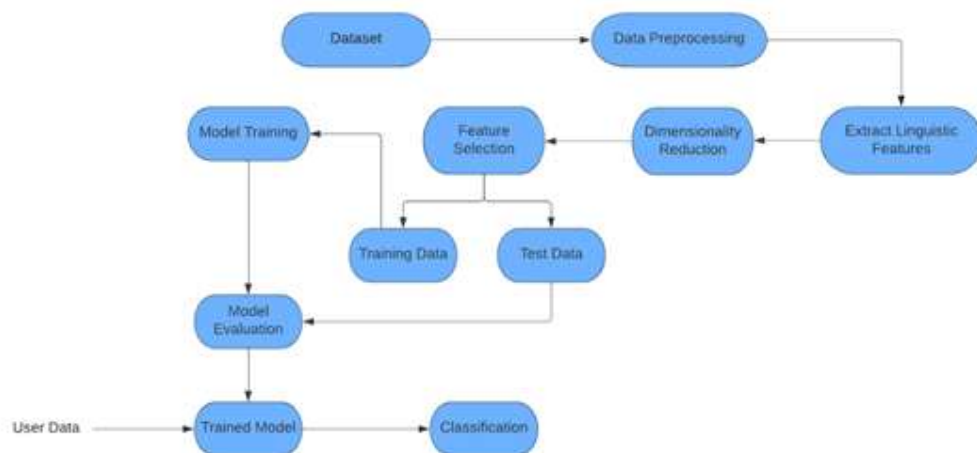
Fig 1. Flow Chart

Overview of the process:

The following is an overview of the process of building a house price prediction model by feature selection, model training and evaluation.

Prepare the data:

This includes cleaning the data, removing outliers, and handling missing values.

Perform feature selection:

This can be done using a variety of methods, such as correlation analysis, information gain, and recursive feature elimination.

Train the model:

The first step of NLP model training is to collect and prepare the data that the model will use to learn from. Depending on the task and the language, you may need different types and sources of data, such as text, audio, or images.

Evaluate the model:

There are two ways to evaluate language models in NLP:

Extrinsic evaluation

Intrinsic evaluation.

Intrinsic evaluation captures how well the model captures what it is supposed to capture, like probabilities.

Extrinsic evaluation (or task-based evaluation) captures how useful the model is in a particular task.

Deploy the model:

Model deployment involves making the model available to end users. Once the model is trained and validated, it is ready for deployment.

Real:

	title	text	subject	date
0	As U.S. budget fight looms, Republicans flip t...	WASHINGTON (Reuters) - The head of a conservat...	politicsNews	December 31, 2017
1	U.S. military to accept transgender recruits o...	WASHINGTON (Reuters) - Transgender people will...	politicsNews	December 29, 2017
2	Senior U.S. Republican senator: 'Let Mr. Muell...	WASHINGTON (Reuters) - The special counsel inv...	politicsNews	December 31, 2017
3	FBI Russia probe helped by Australian diplomat...	WASHINGTON (Reuters) - Trump campaign adviser ...	politicsNews	December 30, 2017
4	Trump wants Postal Service to charge 'much mor...	SEATTLE/WASHINGTON (Reuters) - President Donal...	politicsNews	December 29, 2017

Fake:

	title	text	subject	date
0	Donald Trump Sends Out Embarrassing New Year'...	Donald Trump just couldn t wish all Americans ...	News	December 31, 2017
1	Drunk Bragging Trump Staffer Started Russian ...	House Intelligence Committee Chairman Devin Nu...	News	December 31, 2017
2	Sheriff David Clarke Becomes An Internet Joke...	On Friday, it was revealed that former Milwauk...	News	December 30, 2017
3	Trump Is So Obsessed He Even Has Obama's Name...	On Christmas day, Donald Trump announced that ...	News	December 29, 2017
4	Pope Francis Just Called Out Donald Trump Dur...	Pope Francis used his annual Christmas Day mes...	News	December 25, 2017

Feature selection:

Feature Selection is the process where you automatically or manually select those features which contribute most to your prediction variable or output in which you are interested in.

Having irrelevant features in your data can decrease the accuracy of the models and make your model learn based on irrelevant features.

Motivation:

We want to decide how to *embed* our *documents* into a vector space in a way that takes account of what we think is important about them. This means that we need to formally define which elements of our *documents* are worth taking notice of, and which can be safely ignored.

Removing punctuation:

There are various different ways of doing it but it's always just one line of code, or a parameter set in a function. Despite the practical ease of achieving this, it still needs thought.

You can choose to replace the punctuation marks with a space, or remove them altogether. Replacing them with a space turns *it's* into the two words *it* and *s*, rather than the single word *its*.

Choosing word to keep and drop:

The advantage of removing words from our analysis is twofold. First, by focussing less on words that we aren't interested in, we hope that resulting *embedding* of the *documents* encodes their relevant content rather than irrelevant aspects.

Second, by reducing the size of the *vocabulary* we can help reduce the memory required to store our calculations and hopefully make things runs faster.

Stopwords:

One method of removing words we don't want our model to consider is to create a list of *stopwords* that are always to be ignored.

Typos

Sometimes typos can be frequent enough that you spot them in your results and can correct them as part of feature selection.

For example, in one of pur projects we had several occurrences of the typo *rehabilitaiton*, which meant the *documents* with this word were not

grouped with those containing *rehabilitation*. In that case we were able to correct this typo automatically.

Stemming:

The solution to this is *stemming*, where we follow an algorithm that reduces words down to their roots. In this way *save*, *saved*, *saves*, and *saving* are all reduced to *save*, and *documents* that contain any of these words are automatically seen to have something in common.

<u>Feature Engineering:</u>

Feature engineering is one of the most important steps in machine learning. It is the process of using domain knowledge of the data to create features that make machine learning algorithms work.

Think machine learning algorithm as a learning child the more accurate information you provide the more they will be able to interpret the information well.

NLP is a subfield of artificial intelligence where we understand human interaction with machines using natural languages.

NLP task overview:

To understand the feature engineering task in NLP, we will be implementing it on a Twitter dataset. We will be using <u>Fake News Dataset</u>. The task is to classify the tweet as **Fake** or **Real.** The dataset is divided into train, validation, and test set. Below is the distribution,

1.Number of Characters

Count the number of characters present in a tweet.

```
def count_chars(text):
    return len(text)
```

2. Number of words

Count the number of words present in a tweet.

```
def count_words(text):
    return len(text.split())
```

3. Number of capital characters

Count the number of capital characters present in a tweet.

4. Number of capital words

Count the number of capital words present in a tweet.

```python
def count_capital_words(text):
    return sum(map(str.isupper,text.split()))
```

5. Count the number of punctuations

In this function, we return a dictionary of 32 punctuation with the counts, which can be used as separate features, which I will discuss in the next section.

```python
def count_punctuations(text):
    punctuations='!"#$%&\'()*+,-./:;<=>?@[\]^_`{|}~'
    d=dict()
    for i in punctuations:
        d[str(i)+' count']=text.count(i)
    return d
```

6. Number of words in quotes

The number of words in the single quotation and double quotation.

```python
def count_words_in_quotes(text):
    x = re.findall("'.'|\".\"", text)
    count=0
    if x is None:
        return 0
    else:
        for i in x:
```

```
        t=i[1:-1]
        count+=count_words(t)
    return count
```

Reading train, validation, and test set using pandas.

```
train = pd.read_csv("train.csv")
val = pd.read_csv("validation.csv")
test = pd.read_csv(testWithLabel.csv")

# For this task we will combine the train and validation
dataset and then use
# simple train test split from sklern.
df = pd.concat([train, val])
df.head()
```

```
      id   tweet                                             label

0     1    The CDC currently reports 99031 deaths. In gen... real
1     2    States reported 1121 deaths a small rise from ... real
2     3    Politically Correct Woman (Almost) Uses Pandem... fake
3     4    #IndiaFightsCorona: We have 1524 #COVID testin... real
4     5    Populous states can generate large case counts... real
```

Applying the above-defined feature extraction on train and test set.

```
df['char_count']       =       df["tweet"].apply(lambda
x:count_chars(x))
df['word_count']       =       df["tweet"].apply(lambda
x:count_words(x))
df['sent_count']       =       df["tweet"].apply(lambda
x:count_sent(x))
```

```python
df['capital_char_count']    =    df["tweet"].apply(lambda
x:count_capital_chars(x))
df['capital_word_count']    =    df["tweet"].apply(lambda
x:count_capital_words(x))
df['quoted_word_count']    =    df["tweet"].apply(lambda
x:count_words_in_quotes(x))
df['stopword_count']      =      df["tweet"].apply(lambda
x:count_stopwords(x))
df['unique_word_count']    =    df["tweet"].apply(lambda
x:count_unique_words(x))
df['htag_count']          =          df["tweet"].apply(lambda
x:count_htags(x))
df['mention_count']        =        df["tweet"].apply(lambda
x:count_mentions(x))
df['punct_count']          =          df["tweet"].apply(lambda
x:count_punctuations(x))
df['avg_wordlength'] = df['char_count']/df['word_count']
df['avg_sentlength'] = df['word_count']/df['sent_count']
df['unique_vs_words']                                    =
df['unique_word_count']/df['word_count']
df['stopwords_vs_words']                                  =
df['stopword_count']/df['word_count']
# SIMILARLY YOU CAN APPLY THEM ON TEST SET
```

We will create a DataFrame from the dictionary returned by the "punct_count" function and then merge it with the main dataset.

```python
df_punct = pd.DataFrame(list(df.punct_count))
test_punct = pd.DataFrame(list(test.punct_count))

# Merging pnctuation DataFrame with main DataFrame
df    =    pd.merge(df,    df_punct,    left_index=True,
right_index=True)
```
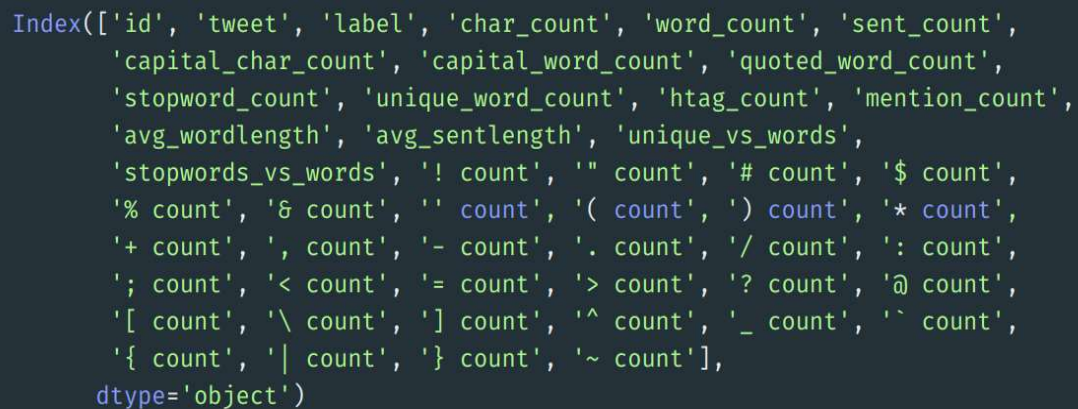
```
test    =    pd.merge(test,    test_punct,left_index=True,
right_index=True)
# We can drop "punct_count" column from both df and
test DataFrame
df.drop(columns=['punct_count'],inplace=True)
test.drop(columns=['punct_count'],inplace=True)
df.columns
```

```
Index(['id', 'tweet', 'label', 'char_count', 'word_count', 'sent_count',
       'capital_char_count', 'capital_word_count', 'quoted_word_count',
       'stopword_count', 'unique_word_count', 'htag_count', 'mention_count',
       'avg_wordlength', 'avg_sentlength', 'unique_vs_words',
       'stopwords_vs_words', '! count', '" count', '# count', '$ count',
       '% count', '& count', '' count', '( count', ') count', '* count',
       '+ count', ', count', '- count', '. count', '/ count', ': count',
       '; count', '< count', '= count', '> count', '? count', '@ count',
       '[ count', '\ count', '] count', '^ count', '_ count', '` count',
       '{ count', '| count', '} count', '~ count'],
      dtype='object')
```
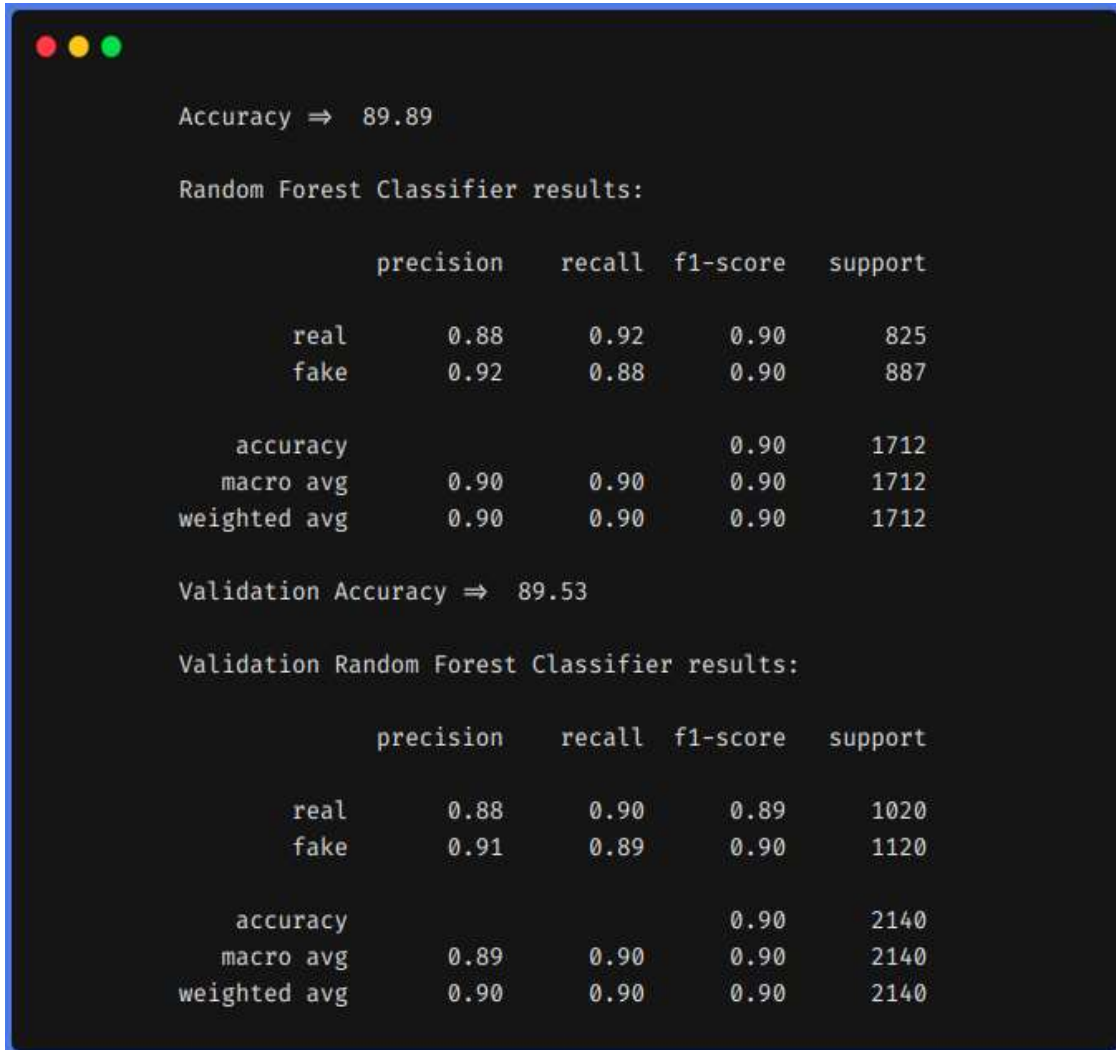
Result comparison:

For comparison, we first trained our model on the above dataset by using features engineering techniques and then without using feature engineering techniques.

In both approaches, we pre-processed the dataset using the same method as described above and TF-IDF was used in both approaches for encoding the text data. You can use whatever

encoding techniques you want to use like word2vec, glove, etc.

1. Without using Feature Engineering techniques

```
Accuracy ⟹  89.89

Random Forest Classifier results:

                precision    recall  f1-score   support

        real        0.88      0.92      0.90       825
        fake        0.92      0.88      0.90       887

    accuracy                            0.90      1712
   macro avg        0.90      0.90      0.90      1712
weighted avg        0.90      0.90      0.90      1712

Validation Accuracy ⟹  89.53

Validation Random Forest Classifier results:

                precision    recall  f1-score   support

        real        0.88      0.90      0.89      1020
        fake        0.91      0.89      0.90      1120

    accuracy                            0.90      2140
   macro avg        0.89      0.90      0.90      2140
weighted avg        0.90      0.90      0.90      2140
```

2. Using Feature Engineering techniques

```
Accuracy ⇒  92.0

Random Forest Classifier results:

              precision    recall  f1-score   support

        real       0.92      0.92      0.92       822
        fake       0.92      0.92      0.92       890

    accuracy                           0.92      1712
   macro avg       0.92      0.92      0.92      1712
weighted avg       0.92      0.92      0.92      1712

Validation Accuracy ⇒  94.07

Validation Random Forest Classifier results:

              precision    recall  f1-score   support

        real       0.93      0.94      0.94      1020
        fake       0.95      0.94      0.94      1120

    accuracy                           0.94      2140
   macro avg       0.94      0.94      0.94      2140
weighted avg       0.94      0.94      0.94      2140
```

From the above results, we can see that feature engineering techniques helped us to increase our **f1** from **0.90 to 0.92** in the train set and from **0.90 to 0.94** in the test set.

Model training:

The first step of NLP model training is to collect and prepare the data that the model will use to learn from. Depending on the task and the language, you may need different types and sources of data, such as text, audio, or images.

Polynomial Recression:

A simple linear regression algorithm only works when the relationship between the data is linear. But suppose we have non-linear data, then linear regression will not be able to draw a best-fit line. Simple regression analysis fails in such conditions.

```python
Import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import r2_score
plt.plot(x_train, lr.predict(x_train), color="r")

plt.plot(X, y, "b.")
```
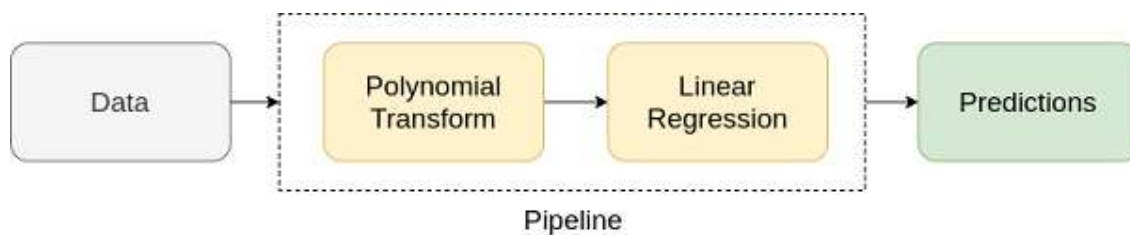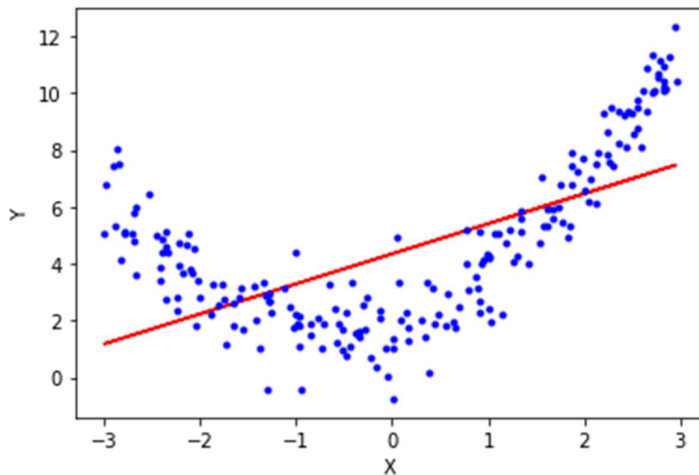
```python
plt.xlabel("X")

plt.ylabel("Y")

plt.show()
```


XGBoost Regressor:

```python
import numpy as np

import pandas as pd

import xgboost as xg
```

```python
from sklearn.model_selection import train_test_split

from sklearn.metrics import mean_squared_error

dataset = pd.read_csv("boston_house.csv")

X, y = dataset.iloc[:, :-1], dataset.iloc[:,

train_X, test_X, train_y, test_y = train_test_split(X, y,

                test_size = 0.3, random_state = 123)

xgb_r = xg.XGBRegressor(objective ='reg:linear',

            n_estimators = 10, seed = 123)


xgb_r.fit(train_X, train_y)


pred = xgb_r.predict(test_X)


rmse = np.sqrt(MSE(test_y, pred))

print("RMSE : % f" %(rmse))
```

Output:

129043.2314

```
# Classifier - Algorithm - SVM
# fit the training dataset on the classifier

SVM = svm.SVC(C=1.0, kernel='linear', degree=3,
gamma='auto')

SVM.fit(Train_X_Tfidf,Train_Y)# predict the labels on validation
dataset

predictions_SVM = SVM.predict(Test_X_Tfidf)

# Use accuracy_score function to get the accuracy

print("SVM Accuracy Score ->
",accuracy_score(predictions_SVM, Test_Y)*100)
```

Output:
SVM Accuracy Score -> 84.6%

Elastic Net:

```
class ElasticRegression() :

    def __init__( self, learning_rate, iterations, l1_penality, l2_penality ) :

    self.learning_rate = learning_rate

     self.iterations = iterations

     self.l1_penality = l1_penality

    self.l2_penality = l2_penality
```

```python
def fit( self, X, Y ) :

    self.m, self.n = X.shape

    self.W = np.zeros( self.n )

    self.b = 0

    self.X = X

    self.Y = Y

    for i in range( self.iterations ) :

        self.update_weights()

    return self

def update_weights( self ) :

    Y_pred = self.predict( self.X )

    dW = np.zeros( self.n )

    for j in range( self.n ) :

        if self.W[j] > 0 :

            dW[j] = ( - ( 2 * ( self.X[:,j] ).dot( self.Y - Y_pred ) ) +

                self.l1_penality + 2 * self.l2_penality * self.W[j] ) / self.m
```

```python
        else :

            dW[j] = ( - ( 2 * ( self.X[:,j] ).dot( self.Y - Y_pred ) )

                      - self.l1_penality + 2 * self.l2_penality * self.W[j] ) / self.

    db = - 2 * np.sum( self.Y - Y_pred ) / self.m

    self.W = self.W - self.learning_rate * dW

    self.b = self.b - self.learning_rate * db

    return self

def predict( self, X ) :

    return X.dot( self.W ) + self.b

def main() :

    df = pd.read_csv( "salary_data.csv"

    X = df.iloc[:,:-1].values

    Y = df.iloc[:,1].values

    X_train, X_test, Y_train, Y_test = train_test_split( X, Y,

                                test_size = 1/3, random_state = 0 )

    model = ElasticRegression( iterations = 1000,
```

```python
                        learning_rate = 0.01, l1_penality = 500, l2_penality = 1 )

    model.fit( X_train, Y_train )

      Y_pred = model.predict( X_test )

    print( "Predicted values ", np.round( Y_pred[:3], 2 ) )

      print( "Real values      ", Y_test[:3] )

      print( "Trained W        ", round( model.W[0], 2 ) )

      print( "Trained b        ", round( model.b, 2 ) )

    plt.scatter( X_test, Y_test, color = 'blue' )

plt.plot( X_test, Y_pred, color = 'orange' )

 plt.title( 'Salary vs Experience' )

 plt.xlabel( 'Years of Experience' )

 plt.ylabel( 'Salary' )

 plt.show()

if __name__ == "__main__" :

 main()
```

Output:
Predicted values [40837.61 122887.43 65079.6]
Real values   [37731 122391 57081]

Trained W   9323.84
Trained b   26851.84

Salary vs Experience

Conclusion:

We have classified our news data using three classification models. We have analysed the performance of the models using accuracy and confusion matrix. But this is only a beginning point for the problem.

There are advanced techniques like BERT, GloVe and ELMo which are popularly used in the field of NLP. If you are interested in NLP, you can work forward with these techniques.