Fake news detection

Koduri Gokul
19BCD7006

```
import pandas as pd
import numpy as np
import·matplotlib.pyplot·as·plt
import·seaborn·as·sns·
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn import feature_extraction, linear_model, model_selection, preprocessing
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
```

Read datasets

```
url = 'https://raw.githubusercontent.com/SushwanthReddy/Fake-News-Detection-using-Machine
```

```
urle = 'https://raw.githubusercontent.com/SushwanthReddy/Fake-News-Detection-using-Machine
```

```
fake = pd.read_csv(url)
true = pd.read_csv(urle)
```

```
fake.shape
```

```
    (23481, 4)
```

```
true.shape
```

```
    (21417, 4)
```

Data cleaning and preparation

```
# Add flag to track fake and real
fake['target'] = 'fake'
true['target'] = 'true'
```

```
# Concatenate dataframes
data = pd.concat([fake, true]).reset_index(drop = True)
data.shape
```

```
(44898, 5)
```

```
# Shuffle the data
from sklearn.utils import shuffle
data = shuffle(data)
data = data.reset_index(drop=True)
```

```
# Check the data
data.head()
```

title text subject date target

Trump looks to deliver would

October 13,
2017

For the moment, U.S.
NATIONS (Reuters)

0In a sudden flurry,

1Trump budget cut bid

for...

true

WASHINGTON (Reuters)
-

make it 'impossible...

24,
2017true

Pr...politicsNews UNITED

March 19,

- U.S. President Dona...

politicsNews May Lowest L...

News

2Trump's Approval Rating
TANKS To The

Since taking the oath

2017 fake

ofoffice, alleged presi...

```
# Removing the date (we won't use it for the analysis)
data.drop(["date"],axis=1,inplace=True)
data.head()
```

title text subject target

WASHINGTON (Reuters) - For the

0In a sudden flurry, Trump looks to deliver

moment, U.S. Pr...politicsNews true UNITED

NATIONS (Reuters) - U.S. President Dona...

for... 1Trump budget cut bid would make it

politicsNews true Since taking the oath of

'impossible...

office,

alleged presi... News fake

2Trump's Approval Rating TANKS To The
Lowest L...

3Donald Trump Campaign CEO Republicans are great at creating N f k

```
# Removing the title (we will only use the text)
data.drop(["title"],axis=1,inplace=True)
data.head()
```

```
                                                text subject target

0 WASHINGTON (Reuters) - For the moment, U.S. Pr... politicsNews true 1
UNITED NATIONS (Reuters) - U.S. President Dona... politicsNews true 2 Since

taking the oath of office, alleged presi... News fake 3 Republicans are great at

creating controversy ... News fake 4 The video below is a much watch! A young

Donal... politics fake
```

```
# Convert to lowercase

data['text'] = data['text'].apply(lambda x: x.lower())
data.head()
```

```
                                                text subject target

0 washington (reuters) - for the moment, u.s. pr... politicsNews true

1 united nations (reuters) - u.s. president dona... politicsNews true

2 since taking the oath of office, alleged presi... News fake

3 republicans are great at creating controversy ... News fake

4 the video below is a much watch! a young donal... politics fake
```

```
# Remove punctuation

import string

def punctuation_removal(text):
    all_list = [char for char in text if char not in
    string.punctuation] clean_str = ''.join(all_list)
    return clean_str

data['text'] = data['text'].apply(punctuation_removal)


# Check
data.head()
```

```
                                                text subject target

0 washington reuters for the moment us presiden... politicsNews true

1 united nations reuters us president donald tr... politicsNews true

2 since taking the oath of office alleged presid... News fake

3 republicans are great at creating controversy ... News fake

4 the video below is a much watch a young donald... politics fake
```

```
# Removing stopwords
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
stop = stopwords.words('english')

data['text'] = data['text'].apply(lambda x: ' '.join([word for word in x.split() if word n
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data] Unzipping corpora/stopwords.zip.
```

```
data.head()
```

text subject target

0 washington reuters moment us president donald ... politicsNews true **1** united

nations reuters us president donald tru... politicsNews true **2** since taking oath

office alleged president don... News fake **3** republicans great creating

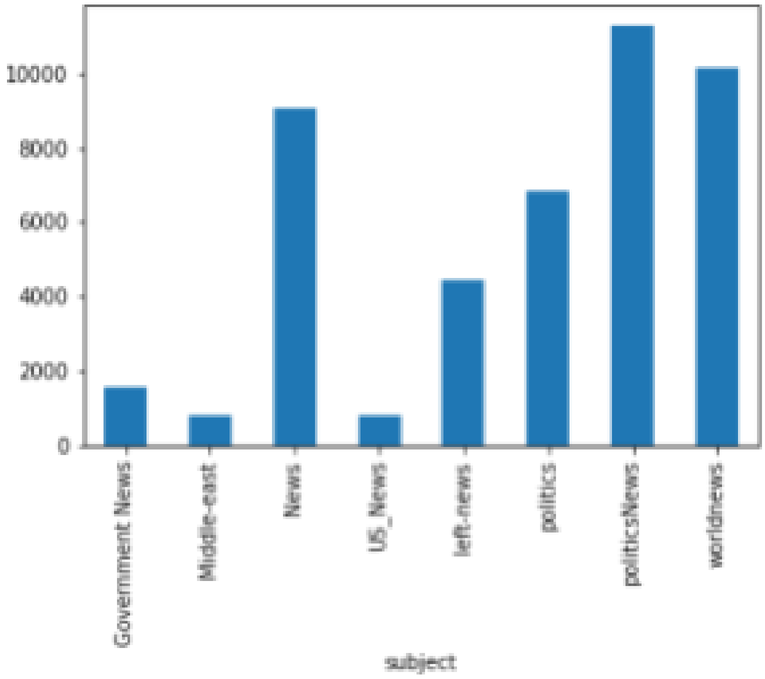
controversy none sh... News fake **4** video much watch young donald j trump

speaks l... politics fake

Basic data exploration

```
# How many articles per subject?
print(data.groupby(['subject'])['text'].count())
data.groupby(['subject'])['text'].count().plot(kind="bar")
plt.show()
```

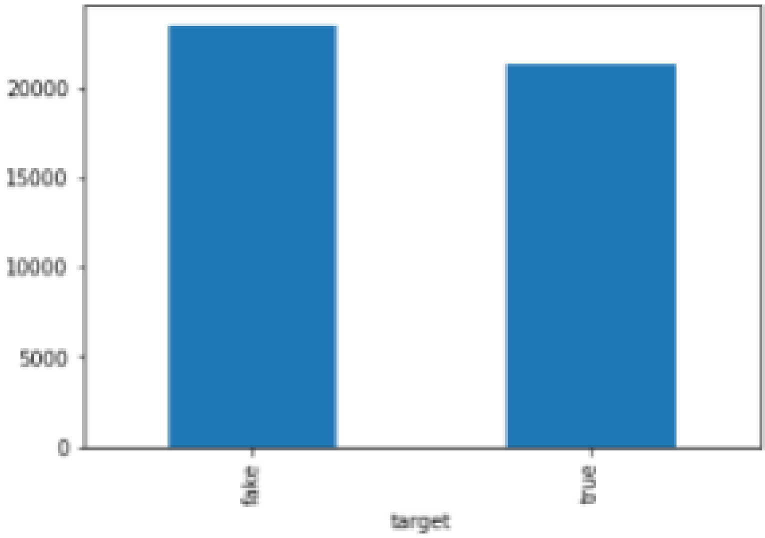
```
subject
Government News 1570
Middle-east 778
News 9050
US_News 783
left-news 4459
politics 6841
politicsNews 11272
worldnews 10145
Name: text, dtype: int64
```



```
# How many fake and real articles?
print(data.groupby(['target'])['text'].count())
```

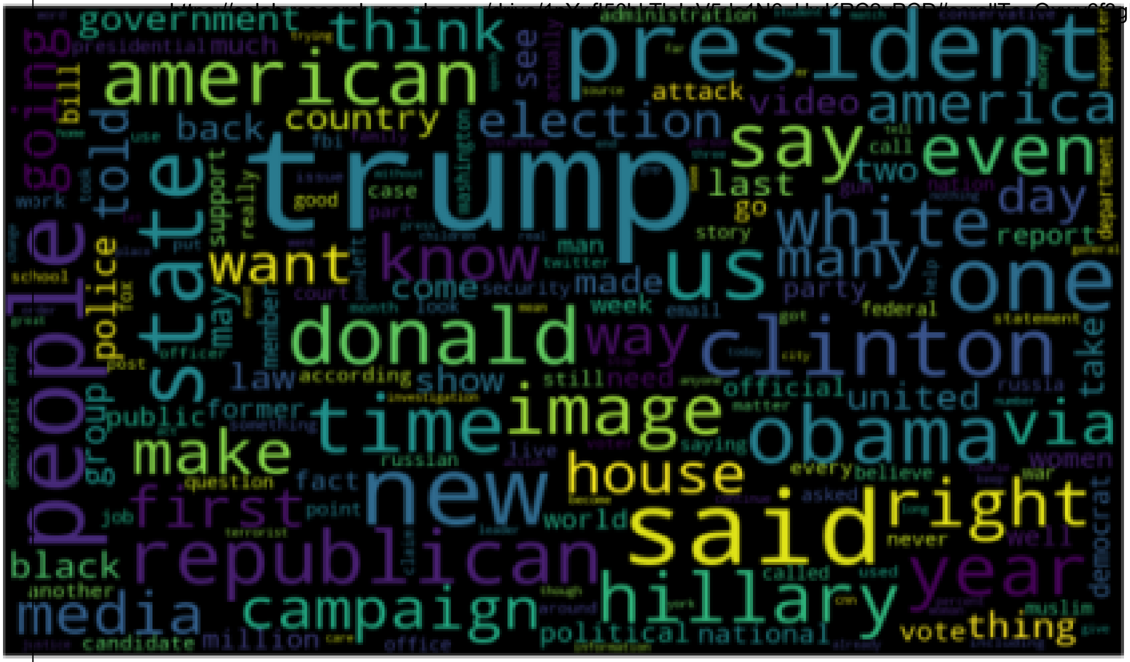
```
data.groupby(['target'])['text'].count().plot(kind="bar")
plt.show()
```

```
target
fake 23481
true 21417
Name: text, dtype: int64
```

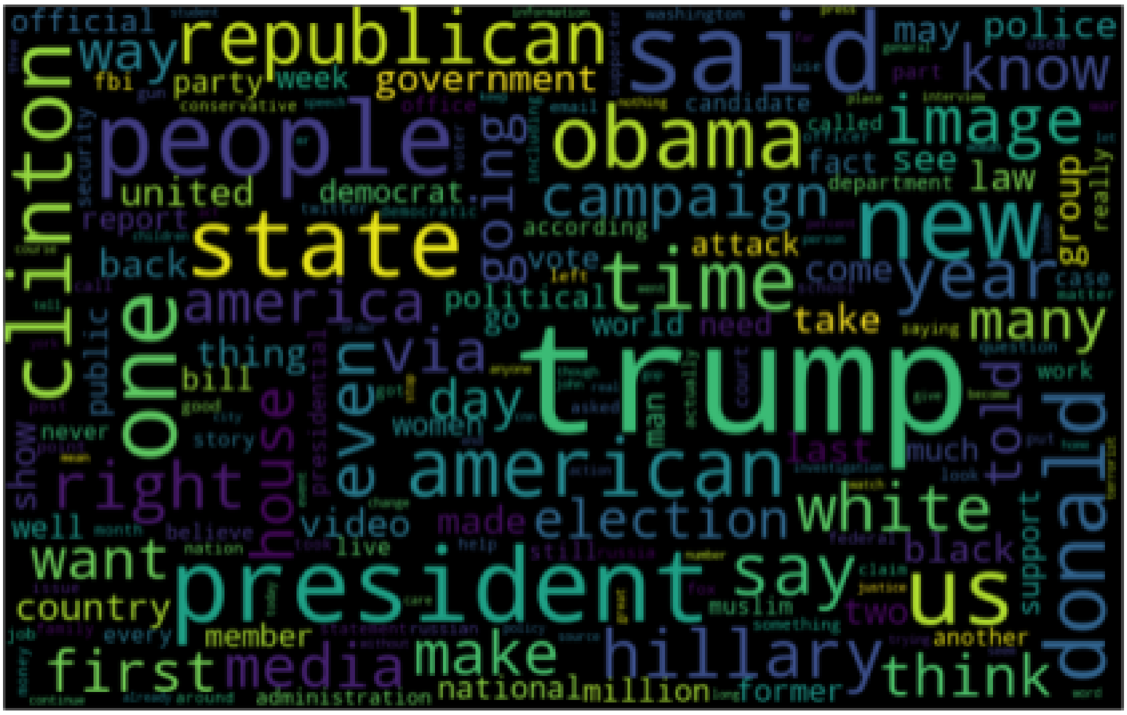


```
# Word cloud for fake news
from wordcloud import WordCloud

fake_data = data[data["target"] == "fake"]
all_words = ' '.join([text for text in fake_data.text])

Word cloud = WordCloud(width= 800, height= 500,
                           max_font_size = 110,
                          collocations = False).generate(all_words)

plt.figure(figsize=(10,7)) plt.imshow(wordcloud,
interpolation='bilinear')
```

```
plt.axis("off")
plt.show()
```

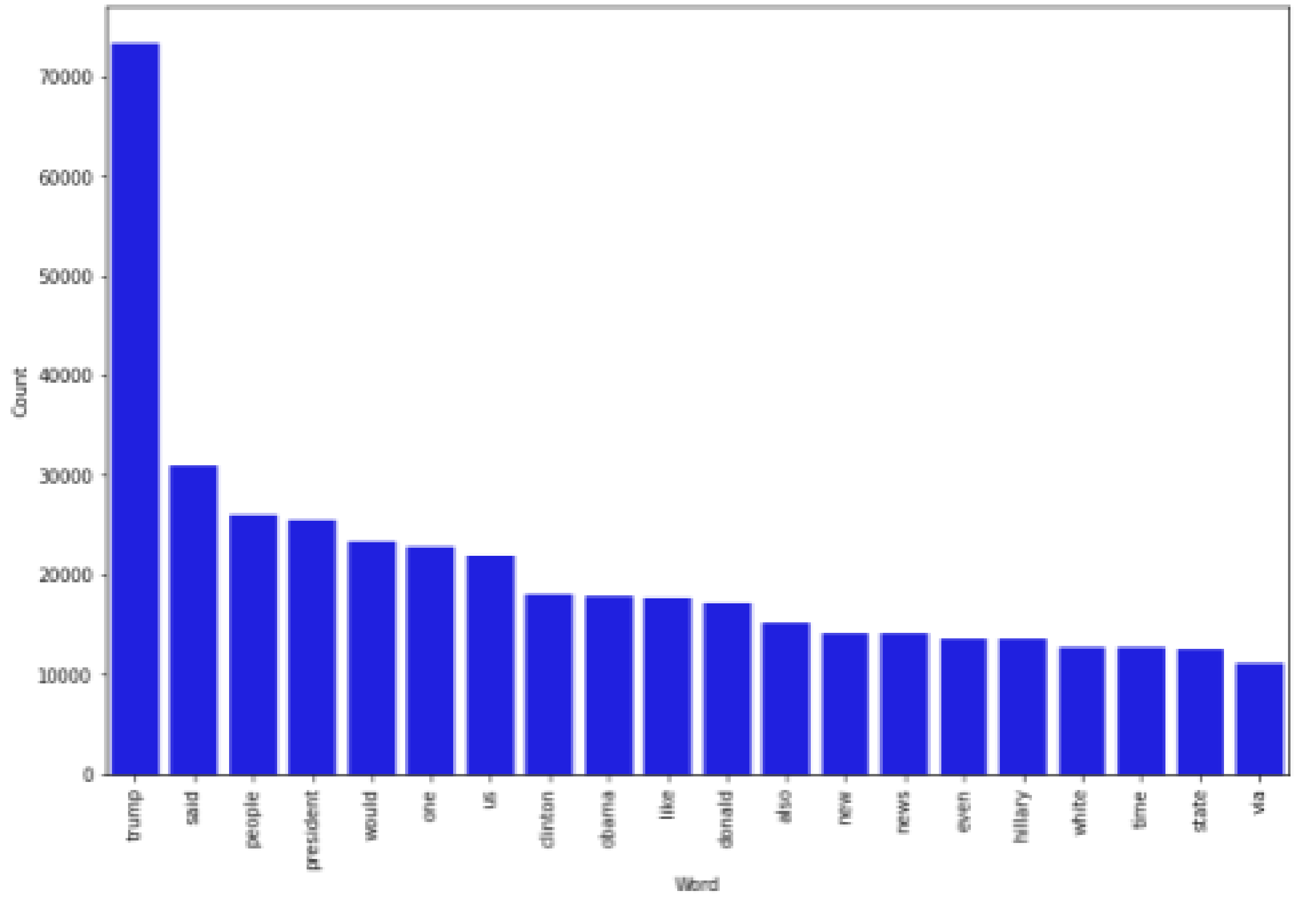


```
# Word cloud for real
        news
        from wordcloud
        import
        WordCloud

        real_data =

data[data["target"] == "true"]
all_words = ' '.join([text for text in fake_data.text])

Word cloud = WordCloud(width= 800, height= 500,
                          max_font_size = 110,
                           collocations = False).generate(all_words)

plt.figure(figsize=(10,7)) plt.imshow(wordcloud,
interpolation='bilinear')plt.axis("off")
plt.show()
```



```
# Most
frequent words counter (Code adapted from https://www.kaggle.com/rodolfoluna/fake-n from
nltk import tokenize

token_space = tokenize.WhitespaceTokenizer()

def counter(text, column_text, quantity):
    all_words = ' '.join([text for text in text[column_text]])
    token_phrase = token_space.tokenize(all_words)
    frequency = nltk.FreqDist(token_phrase)
    df_frequency = pd.DataFrame({"Word": list(frequency.keys()),
                                   "Frequency": list(frequency.values())})
    df_frequency = df_frequency.nlargest(columns = "Frequency", n = quantity)
    plt.figure(figsize=(12,8))
    ax = sns.barplot(data = df_frequency, x = "Word", y = "Frequency", color = 'blue')
    ax.set(ylabel = "Count")
    plt.xticks(rotation='vertical')plt.show()
```

```
# Most frequent words in fake news
counter(data[data["target"] == "fake"], "text", 20)
```



```
# Most frequent words in real news
counter(data[data["target"] == "true"], "text", 20)
```

Modeling

```
# Function to plot the confusion matrix (code from https://scikit-learn.org/stable/auto_ex
from sklearn import metrics
import itertools
```

```
    def plot_confusion_matrix(cm, classes,
                              normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
```

Preparing the data

```
# Split the data
X_train,X_test,y_train,y_test = train_test_split(data['text'], data.target, test_size=0.2,
```

Decision Tree

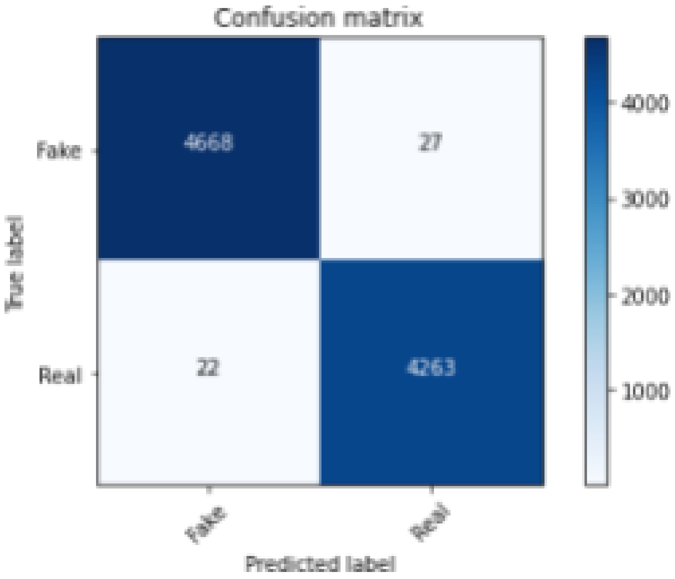
```
from sklearn.tree import DecisionTreeClassifier
dct=dict()
# Vectorizing and applying TF-IDF
pipe = Pipeline([('vect', CountVectorizer()),
                 ('tfidf', TfidfTransformer()),
                 ('model', DecisionTreeClassifier(criterion= 'entropy',
                                            max_depth = 20,
                                            splitter='best',
                                            random_state=42))])
# Fitting the model
model = pipe.fit(X_train, y_train)

# Accuracy
prediction = model.predict(X_test)
```

```
print("accuracy: {}%".format(round(accuracy_score(y_test,
prediction)*100,2))) dct['Decision Tree'] = round(accuracy_score(y_test,
prediction)*100,2)
```

```
accuracy: 99.45%
```

```
cm = metrics.confusion_matrix(y_test, prediction)
plot_confusion_matrix(cm, classes=['Fake', 'Real'])
```

```
Confusion matrix, without normalization
```



Random Forest

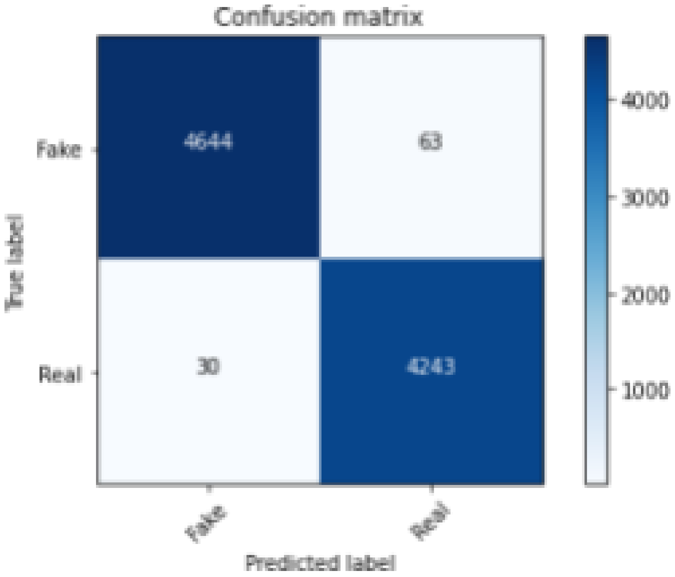
```
from sklearn.ensemble import RandomForestClassifier

pipe = Pipeline([('vect', CountVectorizer()),
                 ('tfidf', TfidfTransformer()),
```

```
                 ('model', RandomForestClassifier(n_estimators=50, criterion="entropy"))])

model = pipe.fit(X_train, y_train)
prediction = model.predict(X_test)
print("accuracy: {}%".format(round(accuracy_score(y_test,
prediction)*100,2))) dct['Random Forest'] = round(accuracy_score(y_test,
prediction)*100,2)
```

```
accuracy: 99.12%
```

```
cm = metrics.confusion_matrix(y_test, prediction)
plot_confusion_matrix(cm, classes=['Fake', 'Real'])
```

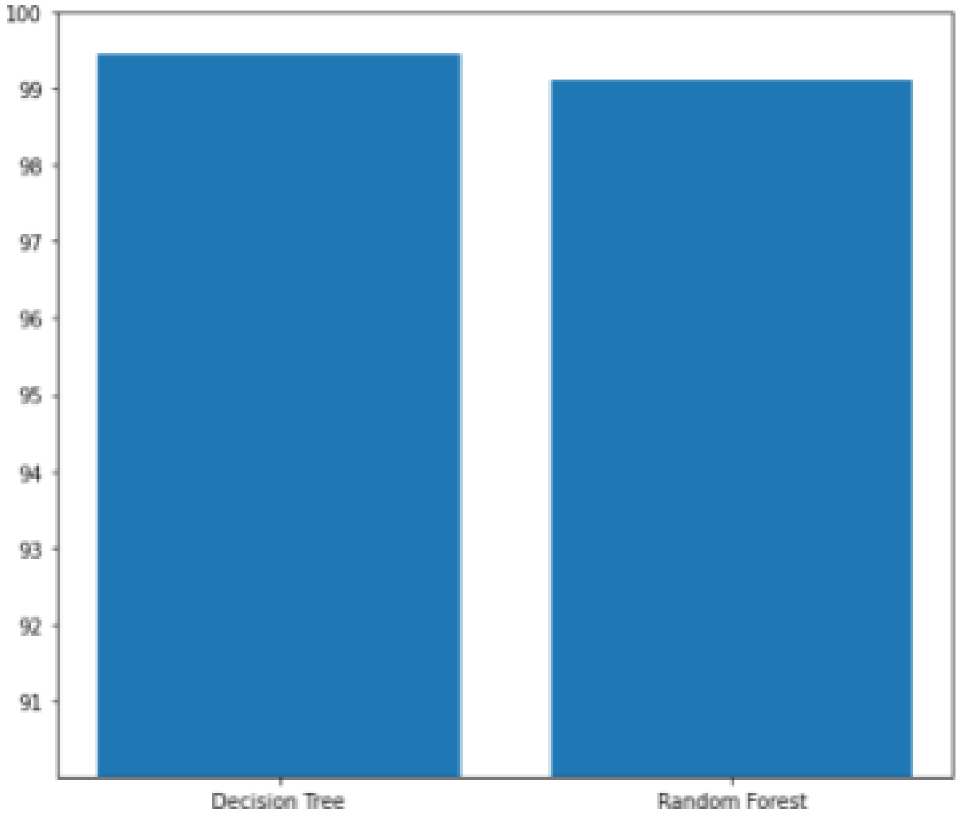
```
Confusion matrix, without normalization
```



Comparing Different Models

```
import matplotlib.pyplot as plt
plt.figure(figsize=(8,7))
plt.bar(list(dct.keys()),list(dct.values()))
plt.ylim(90,100)
plt.yticks((91, 92, 93, 94, 95, 96, 97, 98, 99, 100))
```

```
([<matplotlib.axis.YTick at 0x7f65442f7b90>,
 <matplotlib.axis.YTick at 0x7f653dbf7450>,
 <matplotlib.axis.YTick at 0x7f6544300610>,
 <matplotlib.axis.YTick at 0x7f6544278090>,
 <matplotlib.axis.YTick at 0x7f6540173990>,
 <matplotlib.axis.YTick at 0x7f653dc0e910>,
 <matplotlib.axis.YTick at 0x7f653dc0e750>,
 <matplotlib.axis.YTick at 0x7f653dc08bd0>,
 <matplotlib.axis.YTick at 0x7f6542fa8810>,
 <matplotlib.axis.YTick at 0x7f6542fa8ad0>],
 <a list of 10 Text major ticklabel objects>)
```



check