Importing the neccesary Libraries

[ ] ↳ *1 cell hidden*

Import Dataset

```
1 from google.colab import files
2 uploaded = files.upload()
```

Choose Files bbc-news-data.csv

- **bbc-news-data.csv**(text/csv) - 5080260 bytes, last modified: 7/31/2020 - 100% done

```
Saving bbc-news-data.csv to bbc-news-data.csv
```

```
1 import pandas as pd                   #Imports the pandas library as pd
2 import numpy as np                    #Imports the numpy library for numerical computing
3
4 data_file="bbc-news-data.csv"
5
6 data = pd.read_csv(data_file,sep='\t')     ## Load the CSV file into a DataFrame
```

```
1 data.head(10)
```

	category	filename	title	content	cleaned_content
0	business	001.txt	Ad sales boost Time Warner profit	Quarterly profits at US media giant TimeWarne...	quarterly profit u medium giant timewarner jum...
1	business	002.txt	Dollar gains on Greenspan speech	The dollar has hit its highest level against ...	dollar hit highest level euro almost three mon...
2	business	003.txt	Yukos unit buyer faces loan claim	The owners of embattled Russian oil giant Yuk...	owner embattled russian oil giant yukos ask bu...
3	business	004.txt	High fuel prices hit BA's profits	British Airways has blamed high fuel prices f...	british airway blamed high fuel price 40 drop ...
4	business	005.txt	Pernod takeover talk lifts Domecq	Shares in UK drinks and food firm Allied Dome...	share uk drink food firm allied domecq risen s...
5	business	006.txt	Japan narrowly escapes recession	Japan's economy teetered on the brink of a te...	japan economy teetered brink technical recessi...
6	business	007.txt	Jobs growth still slow in the US	The US created fewer jobs than expected in Ja...	u created fewer job expected january fall jobs...
7	business	008.txt	India calls for fair trade rules	India, which attends the G7 meeting of seven ...	india attends g7 meeting seven leading industr...
8	business	009.txt	Ethiopia's crop production up 24%	Ethiopia produced 14.27 million tonnes of cro...	ethiopia produced 14 27 million tonne crop 200...
9	business	010.txt	Court rejects $280bn tobacco case	A US government claim accusing the country's ...	u government claim accusing country biggest to...

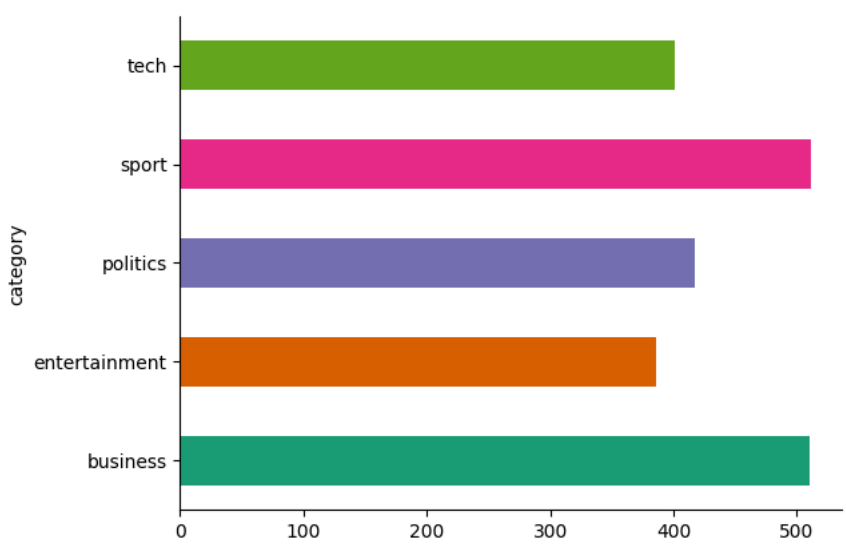
Next steps: Generate code with `data` View recommended plots

```
1 data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2225 entries, 0 to 2224
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   category         2225 non-null   object
 1   filename         2225 non-null   object
 2   title            2225 non-null   object
 3   content          2225 non-null   object
 4   cleaned_content  2225 non-null   object
dtypes: object(5)
memory usage: 87.0+ KB
```

category

```
1
2
3 # @title category
4
5 from matplotlib import pyplot as plt
6 import seaborn as sns
7 data.groupby('category').size().plot(kind='barh', color=sns.palettes.mpl_palette('Dark2'))
8 plt.gca().spines[['top', 'right',]].set_visible(False)
```



Text cleaning/Stop Words Removal

Text Cleaning without stemming/stop word removal

```
#import re                            #Import module for regular expressions

# Text cleaning function
#def clean_text(text):
#   text = re.sub(r'\s+', ' ', text)  # to remove extra spaces, newlines, and tabs.
#   text = re.sub(r'\W', ' ', text)   # to replace all non-word characters with a space.
#  return text.strip().lower()       # to remove any leading and trailing whitespace from the text and Converts the text to lowercase.
```

Text Cleaning using stemming/stop word removal

```
# Text cleaning function
def clean_text(text):
    text = re.sub(r'\W', ' ', text)
    text = text.lower()
    text = re.sub(r'\s+', ' ', text)
    stop_words = set(stopwords.words('english'))
    lemmatizer = WordNetLemmatizer()
    #text = ' '.join([word for word in text.split() if word not in stop_words])
    text = ' '.join([lemmatizer.lemmatize(word) for word in text.split() if word not in stop_words])
    return text


# Clean the content
data['cleaned_content'] = data['content'].apply(clean_text)
```

TFIDF Vectorization

Max Features in Content(text) Column Unigram

```
# Extract the content column
content = data['content']

# Use CountVectorizer to count unique terms
vectorizer = CountVectorizer(stop_words='english')
X_counts = vectorizer.fit_transform(content)

# Get the number of unique terms
num_unique_terms = len(vectorizer.get_feature_names_out())

print(f'Number of unique terms (Unigrams): {num_unique_terms}')
```

```
Number of unique terms (Unigrams): 28980
```

```
# Vectorization
tfidf_vectorizer = TfidfVectorizer(max_features=5000)
X = tfidf_vectorizer.fit_transform(data['cleaned_content'])
y = data['category']
```

Splitting data into Train-test subset

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)


# Check the distribution of the categories in the test set
print("Category distribution in test set:")
print(y_test.value_counts())
```

```
Category distribution in test set:
category
business         142
sport            122
tech             104
politics          95
entertainment     94
Name: count, dtype: int64
```

Model Training and Evaluation

```
from sklearn.metrics import accuracy_score, classification_report, precision_score, recall_score, confusion_matrix
import seaborn as sns

def train_and_evaluate_model(model, X_train, y_train, X_test, y_test):
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    report = classification_report(y_test, y_pred, target_names=y_test.unique())
    cm = confusion_matrix(y_test, y_pred, labels=y_test.unique())
    return accuracy, report, cm, y_pred

# Calculate and print TP, FP, FN, TN
def calculate_metrics(cm):
    for i in range(len(cm)):
        tp = cm[i, i]
        fp = cm[:, i].sum() - tp
        fn = cm[i, :].sum() - tp
        tn = cm.sum() - (tp + fp + fn)
        print(f'Class {i}: TP: {tp}, FP: {fp}, TN: {tn}, FN: {fn}')

def plot_confusion_matrix(cm, classes):
    plt.figure(figsize=(10, 7))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=classes, yticklabels=classes)
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    plt.title('Confusion Matrix')
    plt.show()

# Cross-validation
def cross_validate_model(model, X, y, cv=5):
    scores = cross_val_score(model, X, y, cv=cv)
    print(f"Cross-Validation Scores: {scores}")
    print(f"Mean Accuracy: {scores.mean()}")
    print(f"Standard Deviation: {scores.std()}")
```

Naive Bayes

```
nb_model = MultinomialNB()
nb_accuracy, nb_report, nb_cm, nb_y_pred = train_and_evaluate_model(
    nb_model, X_train, y_train, X_test, y_test)
```

Evaluation_nb

```
# Print results
print("Naive Bayes Classifier:")
print("Accuracy:", nb_accuracy)
print("Classification Report:\n", nb_report)
```

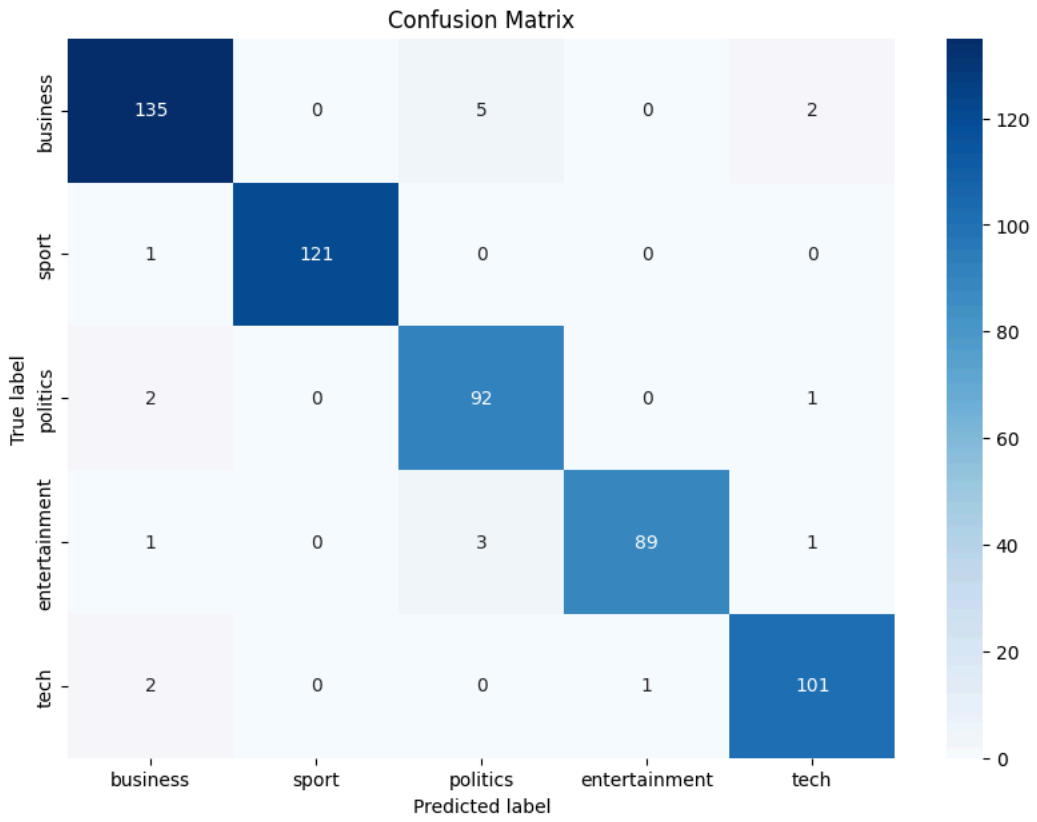
```
Naive Bayes Classifier:
Accuracy: 0.9658886894075404
Classification Report:
                 precision    recall  f1-score   support

       business       0.96      0.95      0.95       142
          sport       0.99      0.95      0.97        94
       politics       0.92      0.97      0.94        95
  entertainment       1.00      0.99      1.00       122
           tech       0.96      0.97      0.97       104

       accuracy                           0.97       557
```

```
macro avg       0.97      0.97      0.97       557
weighted avg       0.97      0.97      0.97       557
```

```
1
2 print("\nMetrics per Class:")
3 calculate_metrics(nb_cm)
```

```
Metrics per Class:
Class 0: TP: 135, FP: 6, TN: 409, FN: 7
Class 1: TP: 121, FP: 0, TN: 435, FN: 1
Class 2: TP: 92, FP: 8, TN: 454, FN: 3
Class 3: TP: 89, FP: 1, TN: 462, FN: 5
Class 4: TP: 101, FP: 4, TN: 449, FN: 3
```

```
1 # Plot the confusion matrix
2 plot_confusion_matrix(nb_cm, classes=y_test.unique())
```



Cross Validation_nb

```
1 print("Naive Bayes Classifier (Cross-Validation):")
2 cross_validate_model(nb_model, X, y)
```

```
Naive Bayes Classifier (Cross-Validation):
Cross-Validation Scores: [0.96179775 0.95730337 0.93932584 0.98202247 0.98202247]
Mean Accuracy: 0.964494382022472
Standard Deviation: 0.016167286066161763
```

Decision Tree

```
1 X_train_dense = X_train.toarray()
2 X_test_dense = X_test.toarray()
```

```
1 # Decision Tree
2 dt_model = DecisionTreeClassifier(random_state=42)
3 dt_accuracy, dt_report, dt_cm, dt_y_pred = train_and_evaluate_model(
4     dt_model, X_train, y_train, X_test, y_test)
```

Evaluation_dt

```
1 print("Decision Tree Classifier:")
2 print("Accuracy:", dt_accuracy)
3 print("Classification Report:\n", dt_report)
```

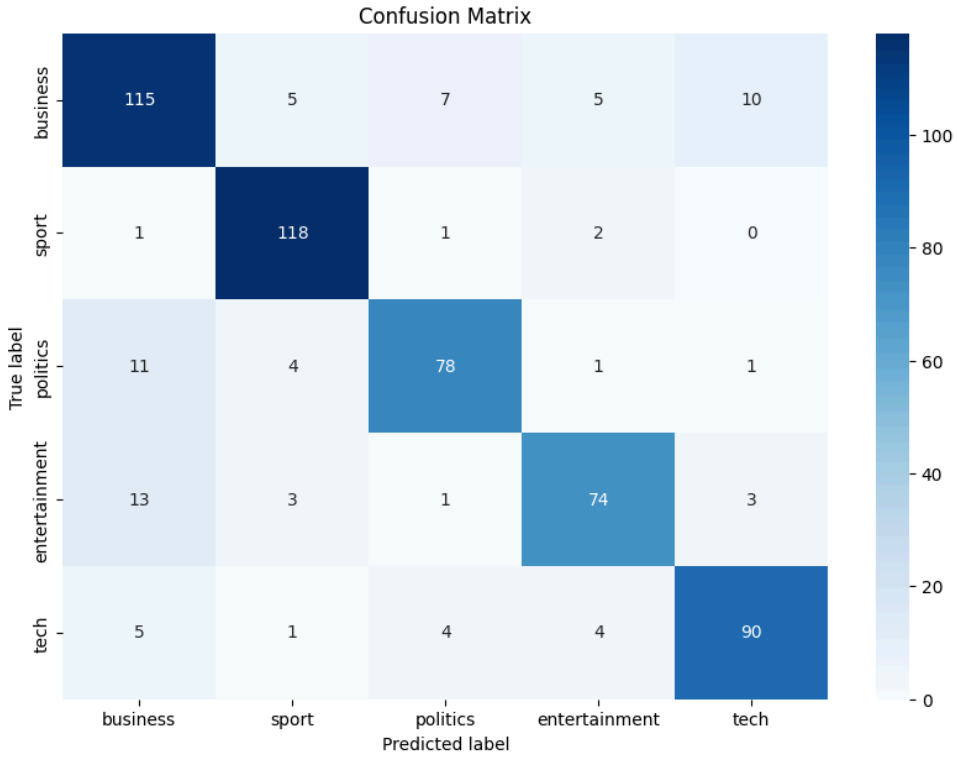
```
Decision Tree Classifier:
Accuracy: 0.8527827648114902
Classification Report:
```

```
                precision    recall  f1-score   support

      business       0.79      0.81      0.80       142
         sport       0.86      0.79      0.82        94
      politics       0.86      0.82      0.84        95
 entertainment       0.90      0.97      0.93       122
          tech       0.87      0.87      0.87       104

      accuracy                           0.85       557
     macro avg       0.86      0.85      0.85       557
  weighted avg       0.85      0.85      0.85       557
```

```
print("\nMetrics per Class:")
calculate_metrics(dt_cm)
```

```
Metrics per Class:
Class 0: TP: 115, FP: 30, TN: 385, FN: 27
Class 1: TP: 118, FP: 13, TN: 422, FN: 4
Class 2: TP: 78, FP: 13, TN: 449, FN: 17
Class 3: TP: 74, FP: 12, TN: 451, FN: 20
Class 4: TP: 90, FP: 14, TN: 439, FN: 14
```

```
# Plot the confusion matrix
plot_confusion_matrix(dt_cm, classes=y_test.unique())
```



Cross Validation_dt

```
print("Decision Tree Classifier (Cross-Validation):")
cross_validate_model(dt_model, X, y)
```

```
Decision Tree Classifier (Cross-Validation):
Cross-Validation Scores: [0.77078652 0.81123596 0.78876404 0.80449438 0.86741573]
Mean Accuracy: 0.8085393258426967
Standard Deviation: 0.032577294875362
```

Random Forest

```
# Random Forest
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_accuracy, rf_report, rf_cm, rf_y_pred = train_and_evaluate_model(
    rf_model, X_train, y_train, X_test, y_test)
```

Evaluation-rf

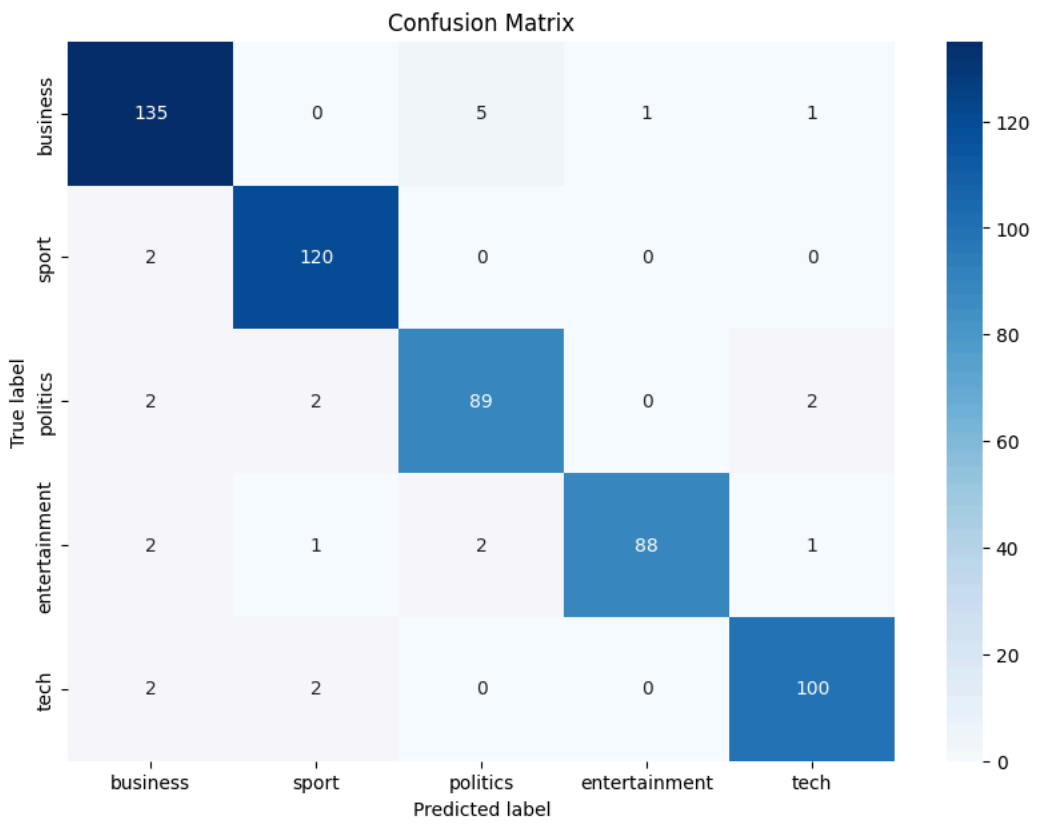
```
print("Random Forest Classifier:")
print("Accuracy:", rf_accuracy)
print("Classification Report:\n", rf_report)
```

```
Random Forest Classifier:
Accuracy: 0.9551166965888689
Classification Report:
                precision    recall  f1-score   support

      business       0.94      0.95      0.95       142
         sport       0.99      0.94      0.96        94
      politics       0.93      0.94      0.93        95
 entertainment       0.96      0.98      0.97       122
          tech       0.96      0.96      0.96       104

      accuracy                           0.96       557
     macro avg       0.96      0.95      0.95       557
  weighted avg       0.96      0.96      0.96       557
```

```
print("\nMetrics per Class:")
calculate_metrics(rf_cm)
```

```
Metrics per Class:
Class 0: TP: 135, FP: 8, TN: 407, FN: 7
Class 1: TP: 120, FP: 5, TN: 430, FN: 2
Class 2: TP: 89, FP: 7, TN: 455, FN: 6
Class 3: TP: 88, FP: 1, TN: 462, FN: 6
Class 4: TP: 100, FP: 4, TN: 449, FN: 4
```

```
# Plot the confusion matrix
plot_confusion_matrix(rf_cm, classes=y_test.unique())
```



Cross Validation_rf

```
print("Random Forest Classifier (Cross-Validation):")
cross_validate_model(rf_model, X, y)

```

```
Random Forest Classifier (Cross-Validation):
Cross-Validation Scores: [0.94606742 0.95955056 0.92359551 0.96629213 0.97303371]
Mean Accuracy: 0.9537078651685393
Standard Deviation: 0.01749925603050048
```

Comparison

```python
# Store the results in a dictionary
results = {
    "Model": ["Naive Bayes", "Decision Tree", "Random Forest"],
    "Accuracy": [nb_accuracy, dt_accuracy, rf_accuracy],
    "Classification Report": [nb_report, dt_report, rf_report]
}

# Convert the results dictionary to a DataFrame for better visualization
results_df = pd.DataFrame(results)
print("Comparison of Model Accuracies and Reports:")
print(results_df)


# Bar plot for accuracies
plt.figure(figsize=(8, 6))
plt.bar(results_df["Model"], results_df["Accuracy"], color=['blue', 'green', 'red'])
plt.xlabel("Model")
plt.ylabel("Accuracy")
plt.title("Accuracy Comparison of Different Models")
plt.show()

```

```
Comparison of Model Accuracies and Reports:
           Model  Accuracy                              Classification Report
0    Naive Bayes  0.965889                precision    recall  f1-score  ...
1  Decision Tree  0.852783                precision    recall  f1-score  ...
2  Random Forest  0.955117                precision    recall  f1-score  ...
```