nlp-text-classification

January 1, 2024

[153]:
```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.feature_extraction.text import CountVectorizer
from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.naive_bayes import MultinomialNB
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC

from sklearn.metrics import accuracy_score
import sklearn.metrics as metrics
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
```

[154]:
```
data = pd.read_csv('/kaggle/input/usnews/US-Economic-News.csv',␣
 ↪encoding='ISO-8859-1')
data.head()
```

[154]:
```
    _unit_id  _golden _unit_state  _trusted_judgments _last_judgment_at  \
0  842613455    False   finalized                   3     12/5/15 17:48
1  842613456    False   finalized                   3     12/5/15 16:54
2  842613457    False   finalized                   3      12/5/15 1:59
3  842613458    False   finalized                   3      12/5/15 2:19
4  842613459    False   finalized                   3     12/5/15 17:48

   positivity  positivity:confidence relevance  relevance:confidence  \
0         3.0                 0.6400       yes                 0.640
1         NaN                    NaN        no                 1.000
2         NaN                    NaN        no                 1.000
3         NaN                 0.0000        no                 0.675
```

```
4        3.0                 0.3257       yes                 0.640

      articleid     date                                    headline  \
0  wsj_398217788   8/14/91              Yields on CDs Fell in the Latest Week
1  wsj_399019502   8/21/07  The Morning Brief: White House Seeks to Limit …
2  wsj_398284048  11/14/91  Banking Bill Negotiators Set Compromise --- Pl…
3  wsj_397959018   6/16/86  Manager's Journal: Sniffing Out Drug Abusers I…
4  wsj_398838054   10/4/02  Currency Trading: Dollar Remains in Tight Rang…

   positivity_gold  relevance_gold  \
0              NaN             NaN
1              NaN             NaN
2              NaN             NaN
3              NaN             NaN
4              NaN             NaN

                                                text
0  NEW YORK -- Yields on most certificates of dep…
1  The Wall Street Journal Online</br></br>The Mo…
2  WASHINGTON -- In an effort to achieve banking …
3  The statistics on the enormous costs of employ…
4  NEW YORK -- Indecision marked the dollar's ton…
```

```
[155]: data.shape
```

```
[155]: (8000, 15)
```

```
[156]: data["relevance"].value_counts()
```

```
[156]: relevance
       no          6571
       yes         1420
       not sure       9
       Name: count, dtype: int64
```

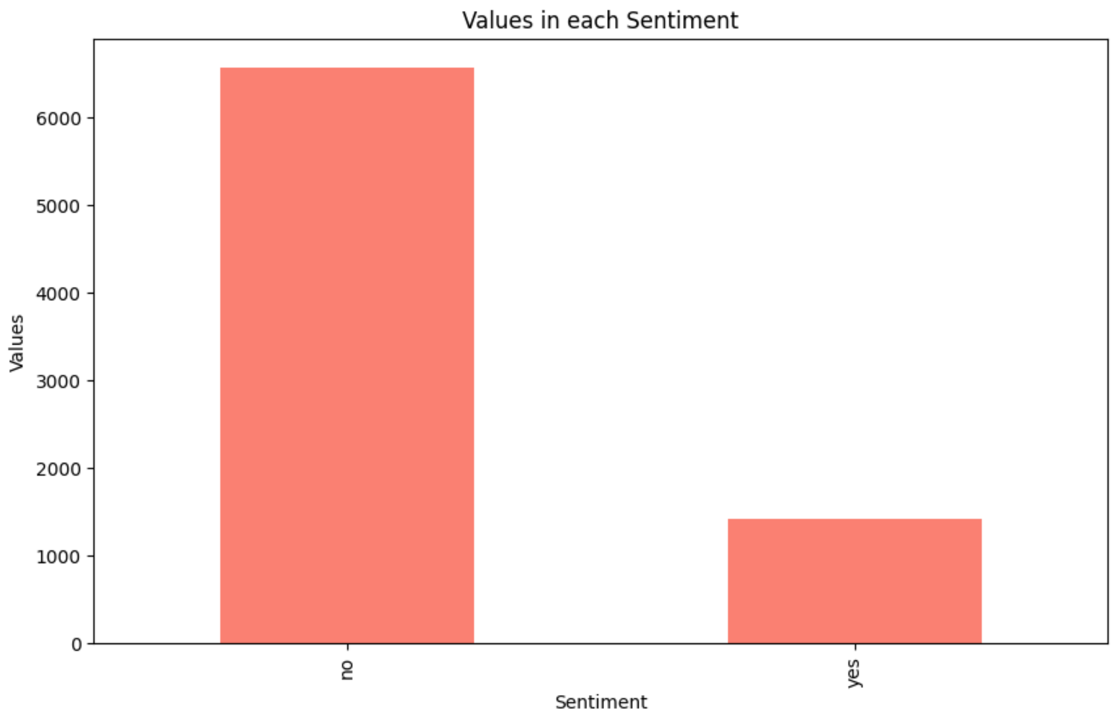
```
[157]: data = data[data.relevance != "not sure"]
       data.shape
```

```
[157]: (7991, 15)
```

```
[158]: data["relevance"].value_counts()/data.shape[0]
```

```
[158]: relevance
       no     0.8223
       yes    0.1777
       Name: count, dtype: float64
```

[159]:
```
plt.figure(figsize=(10, 6))
data['relevance'].value_counts().plot(kind='bar', color='salmon')
plt.title("Values in each Sentiment")
plt.xlabel("Sentiment")
plt.ylabel("Values")
plt.show()
```



1 Text Pre-processing

[160]:
```
data['relevance'] = data.relevance.map({'yes':1, 'no':0}) # relevant is 1,␣
  ↪not-relevant is 0
```

[161]:
```
data = data[["text","relevance"]] # taking text input and output variable as␣
  ↪relevance
data = data[:1000]
data.shape
```

[161]:
```
(1000, 2)
```

[162]:
```
data.head()
```

[162]:
```
                                                text  relevance
0  NEW YORK -- Yields on most certificates of dep…          1
1  The Wall Street Journal Online</br></br>The Mo…          0
2  WASHINGTON -- In an effort to achieve banking …          0
3  The statistics on the enormous costs of employ…          0
4  NEW YORK -- Indecision marked the dollar's ton…          1
```

[163]:
```
data['text'][0]
```

[163]:
```
'NEW YORK -- Yields on most certificates of deposit offered by major banks
dropped more than a tenth of a percentage point in the latest week, reflecting
the overall decline in short-term interest rates.</br></br>On small-
denomination, or "consumer," CDs sold directly by banks, the average yield on
six-month deposits fell to 5.49% from 5.62% in the week ended yesterday,
according to an 18-bank survey by Banxquote Money Markets, a Wilmington, Del.,
information service.</br></br>On three-month "consumer" deposits, the average
yield sank to 5.29% from 5.42% the week before, according to Banxquote. Two
banks in the Banxquote survey, Citibank in New York and CoreStates in
Pennsylvania, are paying less than 5% on threemonth small-denomination
CDs.</br></br>Declines were somewhat smaller on five-year consumer CDs, which
eased to 7.37% from 7.45%, Banxquote said.</br></br>Yields on three-month and
six-month Treasury bills sold at Monday\'s auction plummeted more than a fifth
of a percentage point from the previous week, to 5.46% and 5.63%, respectively.'
```

2 Text Cleaning

2.0.1 Remove named entities.

2.0.2 Convert to lowercase.

2.0.3 Replace “” with a space.

2.0.4 Replace hyphens with spaces.

2.0.5 Remove punctuation and digits.

2.0.6 Remove stopwords.

2.0.7 Apply lemmatization.

[164]:
```
pip install stop-words
```

```
Requirement already satisfied: stop-words in /opt/conda/lib/python3.10/site-
packages (2018.7.23)
Note: you may need to restart the kernel to use updated packages.
```

[165]:
```
import spacy
from stop_words import get_stop_words
import string
```

```
nlp = spacy.load('en_core_web_sm')
stopwords = get_stop_words('en')  # Use 'en' for English, adjust for other␣
 ↪languages

def clean(doc):
    text_no_namedentities = []
    document = nlp(doc)
    ents = [e.text for e in document.ents]
    for item in document:
        if item.text in ents:
            pass
        else:
            text_no_namedentities.append(item.text)
    doc = " ".join(text_no_namedentities)

    doc = doc.lower().strip()
    doc = doc.replace("</br>", " ")
    doc = doc.replace("-", " ")
    doc = "".join([char for char in doc if char not in string.punctuation and␣
 ↪not char.isdigit()])
    doc = " ".join([token for token in doc.split() if token not in stopwords])
    # Use spaCy lemmatizer
    doc = " ".join([token.lemma_ for token in nlp(doc)])
    return doc
```

[166]:
```
# Test the clean function
cleaned_text = clean(data['text'][0])
print(cleaned_text)
```

```
new york yield certificate deposit offer major bank drop tenth percentage point
late week reflect overall decline short term interest rate br small denomination
consumer cd sell directly bank average yield six month deposit fall week end
yesterday accord bank survey money market information service br three month
consumer deposit average yield sink week accord bank survey new york pay less
small denomination decline somewhat small five year consumer cd ease say br
yield three month six month bill sell s auction plummet fifth percentage point
previous week respectively
```

[167]:
```
data['text'] = data['text'].apply(clean)
data.head()
```

[167]:
```
                                                text  relevance
0  new york yield certificate deposit offer major…          1
1  wall street journal online br morning brief lo…          0
2  effort achieve banking reform negotiator admin…          0
3  statistic enormous cost employee drug abuse we…          0
4  new york indecision mark dollar s tone trader …          1
```

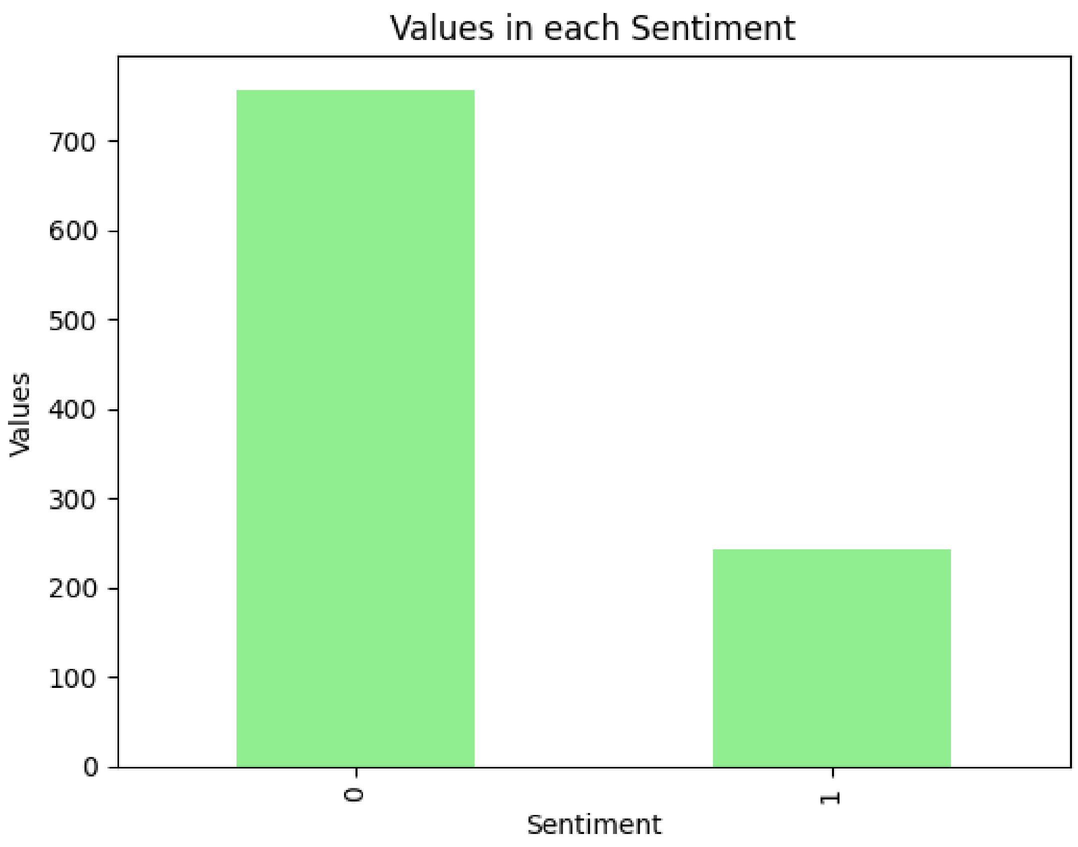
3 TF-IDF Vectorizer

[168]:
```
docs = list(data['text'])
tfidf_vectorizer = TfidfVectorizer(use_idf=True, max_features = 20000)
tfidf_vectorizer_vectors = tfidf_vectorizer.fit_transform(docs)
docs = tfidf_vectorizer_vectors.toarray()
```

[169]:
```
X = docs
y = data['relevance']
print(X.shape, y.shape)
```

```
(1000, 9725) (1000,)
```

[170]:
```
y.value_counts().plot(kind='bar',color="lightgreen")
plt.title("Values in each Sentiment")
plt.xlabel("Sentiment")
plt.ylabel("Values")
plt.show()
```



3.1 Train-Test Split

[171]:
```
X_train,X_test,y_train,y_test=train_test_split(X, y, test_size=0.2,␣
↪random_state=123, stratify=y)
```

[172]:
```
print(X_train.shape, y_train.shape)
print(X_test.shape, y_test.shape)
```

```
(800, 9725) (800,)
(200, 9725) (200,)
```

4 Naive Bayes Classifier

4.1 Gaussian Naive Bayes

[173]:
```
gnb = GaussianNB()
gnb.fit(X_train, y_train)
```

[173]:
```
GaussianNB()
```

[174]:
```
y_pred_train = gnb.predict(X_train)
y_pred_test = gnb.predict(X_test)
```

[175]:
```
print("Training Accuracy score:",accuracy_score(y_train, y_pred_train))
print("Testing Accuracy score:",accuracy_score(y_test, y_pred_test))
```

```
Training Accuracy score: 0.995
Testing Accuracy score: 0.755
```

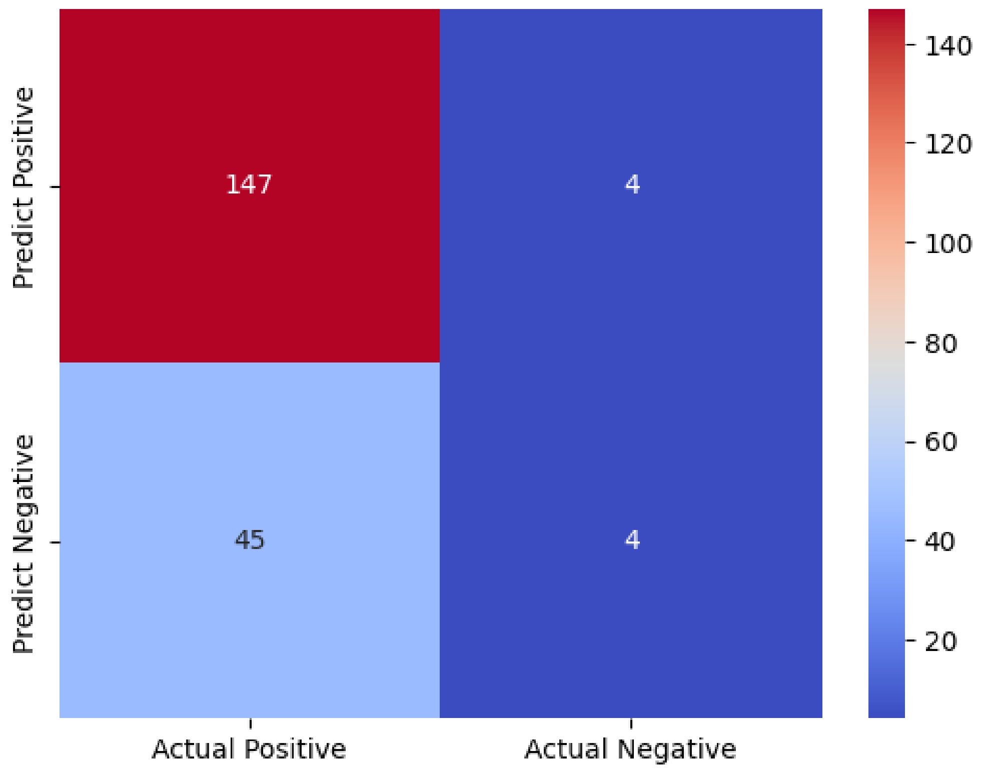
[176]:
```
from sklearn.metrics import classification_report
report = classification_report(y_test, y_pred_test, target_names=['not␣
↪relevant', 'relevant'])
print(report)
```

```
              precision    recall  f1-score   support

not relevant       0.77      0.97      0.86       151
    relevant       0.50      0.08      0.14        49

    accuracy                           0.76       200
   macro avg       0.63      0.53      0.50       200
weighted avg       0.70      0.76      0.68       200
```

[177]:
```
cm = confusion_matrix(y_test, y_pred_test)
cm_matrix = pd.DataFrame(data=cm, columns=['Actual Positive', 'Actual␣
↪Negative'],
                        index=['Predict Positive', 'Predict Negative'])
sns.heatmap(cm_matrix, annot=True, fmt='d', cmap='coolwarm')
plt.show()
```



4.2 Multinomial Naive Bayes

[178]:
```
mnb = MultinomialNB()
mnb.fit(X_train, y_train)
```

[178]:
```
MultinomialNB()
```

[179]:
```
y_pred_train = mnb.predict(X_train)
y_pred_test = mnb.predict(X_test)
```

[180]:
```
print("Training Accuracy score:",accuracy_score(y_train, y_pred_train))
print("Testing Accuracy score:",accuracy_score(y_test, y_pred_test))
```

```
Training Accuracy score: 0.7575
Testing Accuracy score: 0.755
```

[181]:
```
result = classification_report(y_test, y_pred_test, target_names=['not␣
↪relevant', 'relevant'])
print(result)
```

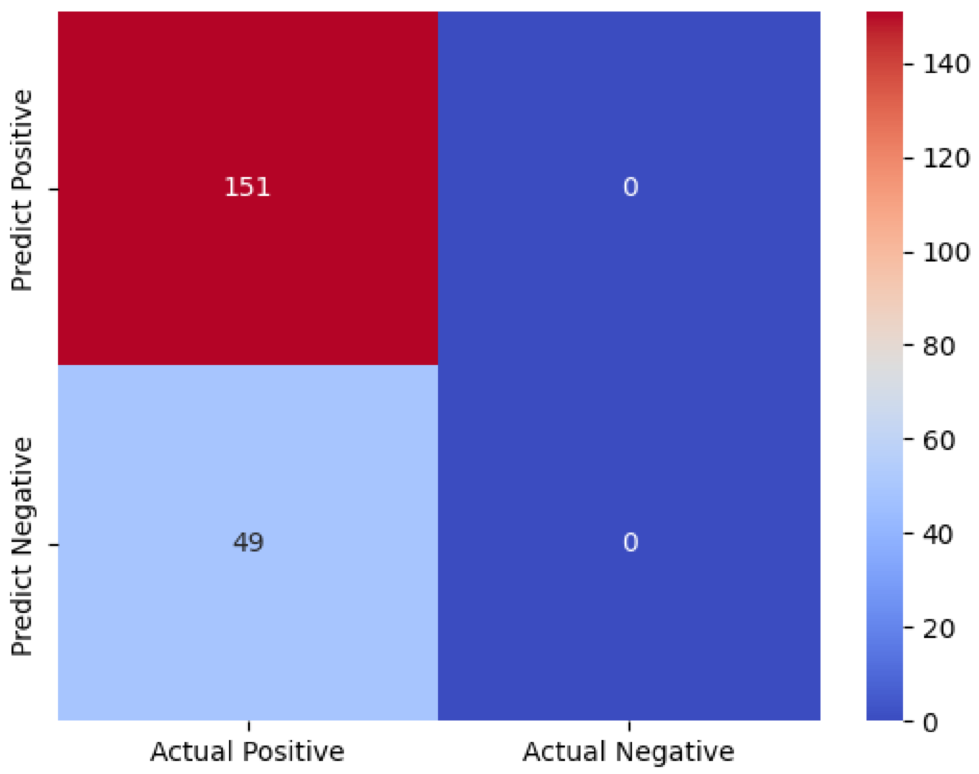
```
              precision    recall  f1-score   support

not relevant       0.76      1.00      0.86       151
    relevant       0.00      0.00      0.00        49

    accuracy                           0.76       200
   macro avg       0.38      0.50      0.43       200
weighted avg       0.57      0.76      0.65       200
```

```
/opt/conda/lib/python3.10/site-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/opt/conda/lib/python3.10/site-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/opt/conda/lib/python3.10/site-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

[182]:
```
cm = confusion_matrix(y_test, y_pred_test)
cm_matrix = pd.DataFrame(data=cm, columns=['Actual Positive', 'Actual␣
↪Negative'],
                        index=['Predict Positive', 'Predict Negative'])
sns.heatmap(cm_matrix, annot=True, fmt='d', cmap='coolwarm')
plt.show()
```



4.3 Logistic Regression Classifier

```
[183]: lr = LogisticRegression(random_state=42)
       lr.fit(X_train, y_train)
```

```
[183]: LogisticRegression(random_state=42)
```

```
[184]: y_pred_train = lr.predict(X_train)
       y_pred_test = lr.predict(X_test)
```

```
[185]: print("Training Accuracy score:",accuracy_score(y_train, y_pred_train))
       print("Testing Accuracy score:",accuracy_score(y_test, y_pred_test))
```

```
Training Accuracy score: 0.815
Testing Accuracy score: 0.765
```

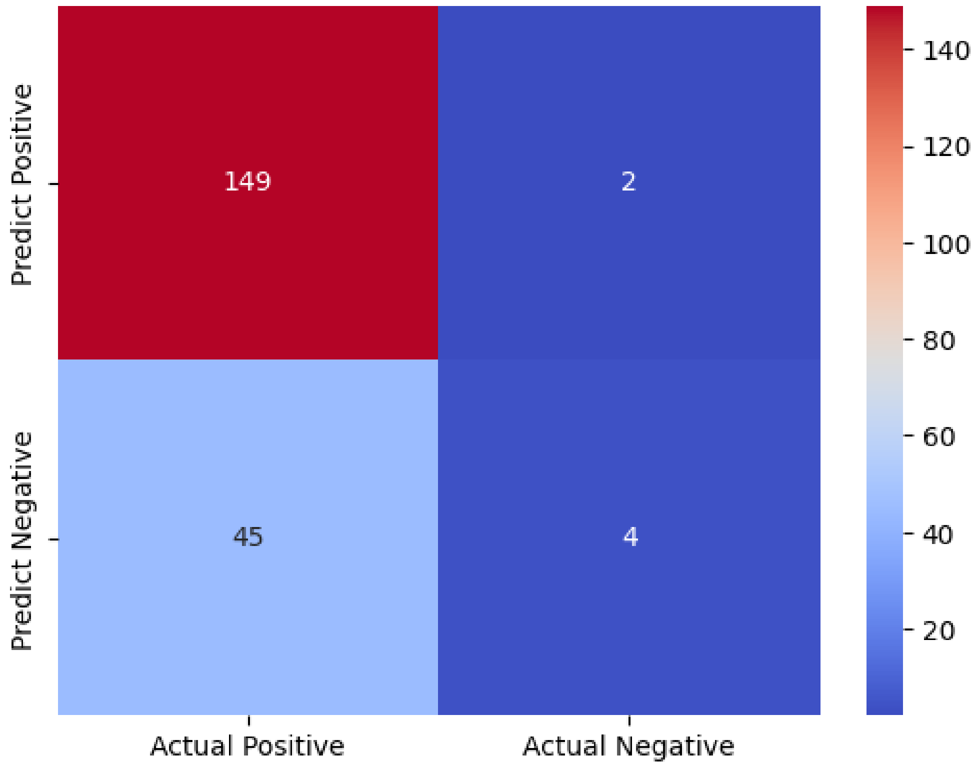
```
[186]: results = classification_report(y_test, y_pred_test, target_names=['not␣
        ↪relevant', 'relevant'])
       print(results)
```

```
              precision    recall  f1-score   support
```

```
not relevant       0.77      0.99      0.86       151
    relevant       0.67      0.08      0.15        49

    accuracy                           0.77       200
   macro avg       0.72      0.53      0.50       200
weighted avg       0.74      0.77      0.69       200
```

[187]:
```
cm = confusion_matrix(y_test, y_pred_test)
cm_matrix = pd.DataFrame(data=cm, columns=['Actual Positive', 'Actual␣
 ↪Negative'],
                        index=['Predict Positive', 'Predict Negative'])
sns.heatmap(cm_matrix, annot=True, fmt='d', cmap='coolwarm')
plt.show()
```



4.4 Support Vector Machines

[188]:
```
svc =  LinearSVC(class_weight='balanced')
svc.fit(X_train, y_train)
```

[188]:
```
LinearSVC(class_weight='balanced')
```

[189]:
```
y_pred_train = svc.predict(X_train)
y_pred_test = svc.predict(X_test)
```

[190]:
```
print("Training Accuracy score:",accuracy_score(y_train, y_pred_train))
print("Testing Accuracy score:",accuracy_score(y_test, y_pred_test))
```

```
Training Accuracy score: 0.99625
Testing Accuracy score: 0.705
```

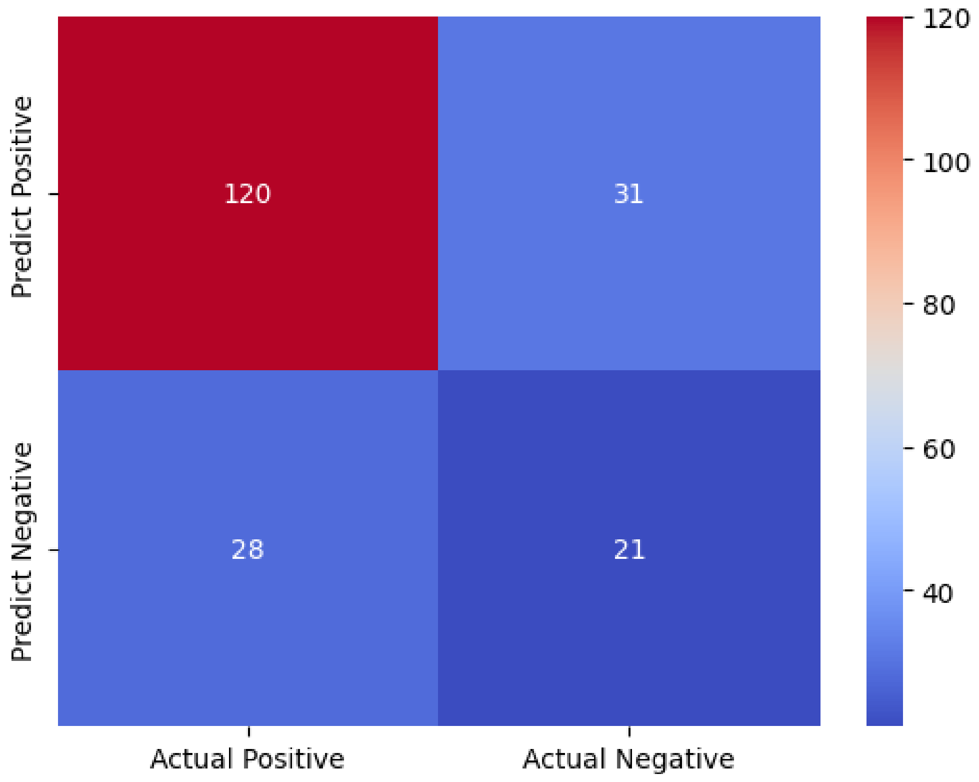
[191]:
```
repo = classification_report(y_test, y_pred_test, target_names=['not relevant',␣
 ↪'relevant'])
print(repo)
```

```
              precision    recall  f1-score   support

not relevant       0.81      0.79      0.80       151
    relevant       0.40      0.43      0.42        49

    accuracy                           0.70       200
   macro avg       0.61      0.61      0.61       200
weighted avg       0.71      0.70      0.71       200
```

[192]:
```
cm = confusion_matrix(y_test, y_pred_test)
cm_matrix = pd.DataFrame(data=cm, columns=['Actual Positive', 'Actual␣
 ↪Negative'],
                        index=['Predict Positive', 'Predict Negative'])
sns.heatmap(cm_matrix, annot=True, fmt='d', cmap='coolwarm')
plt.show()
```



4.5 Decision Tree Classifier

[193]:
```python
from sklearn.tree import DecisionTreeClassifier

dt = DecisionTreeClassifier(random_state=42)
dt.fit(X_train, y_train)
```

[193]:
```
DecisionTreeClassifier(random_state=42)
```

[194]:
```python
y_pred_train = dt.predict(X_train)
y_pred_test = dt.predict(X_test)
print("Training Accuracy score:",accuracy_score(y_train, y_pred_train))
print("Testing Accuracy score:",accuracy_score(y_test, y_pred_test))
```

```
Training Accuracy score: 1.0
Testing Accuracy score: 0.725
```

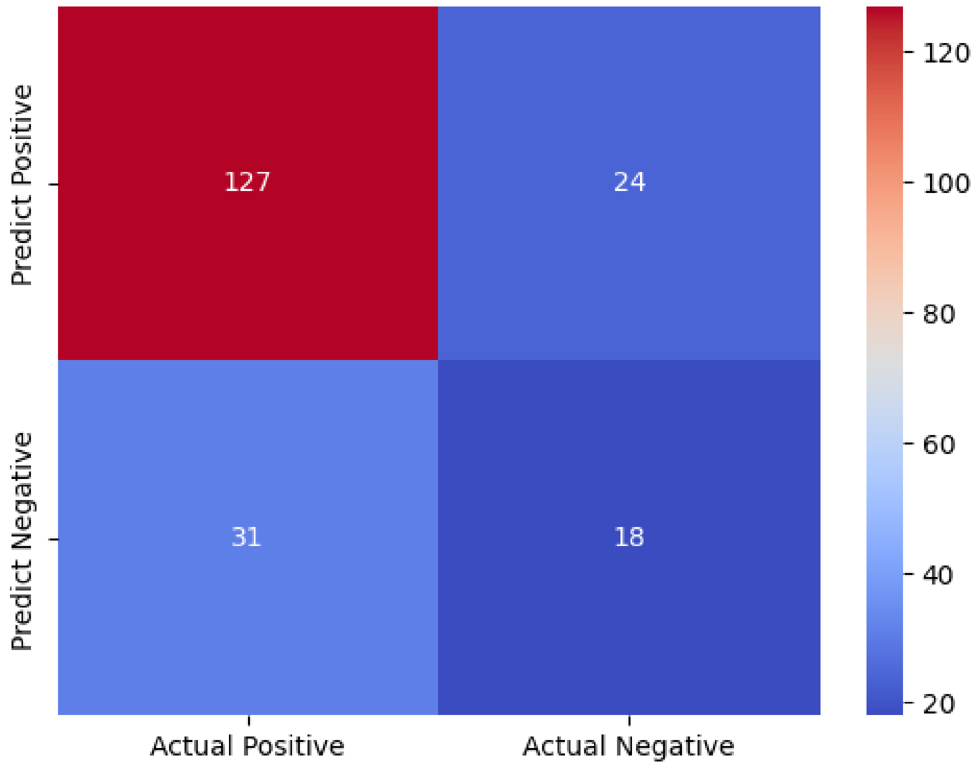
[195]:
```python
repor = classification_report(y_test, y_pred_test, target_names=['not␣
↪relevant', 'relevant'])
print(repor)
```

```
              precision    recall  f1-score   support
```

```
not relevant       0.80      0.84      0.82       151
    relevant       0.43      0.37      0.40        49

    accuracy                           0.73       200
   macro avg       0.62      0.60      0.61       200
weighted avg       0.71      0.72      0.72       200
```

[196]:
```
cm = confusion_matrix(y_test, y_pred_test)
cm_matrix = pd.DataFrame(data=cm, columns=['Actual Positive', 'Actual␣
 ↪Negative'],
                        index=['Predict Positive', 'Predict Negative'])
sns.heatmap(cm_matrix, annot=True, fmt='d', cmap='coolwarm')
plt.show()
```



4.6 Ensembling

[197]:
```
from sklearn.ensemble import VotingClassifier
classifiers = [('Decision Tree', dt),
               ('Logistic Regression', lr),
                ('Naive Bayes', gnb)
               ]
```

[198]:
```
vc = VotingClassifier(estimators=classifiers)
vc.fit(X_train, y_train)
```

[198]:
```
VotingClassifier(estimators=[('Decision Tree',
                              DecisionTreeClassifier(random_state=42)),
                             ('Logistic Regression',
                              LogisticRegression(random_state=42)),
                             ('Naive Bayes', GaussianNB())])
```

[199]:
```
y_pred_train=vc.predict(X_train)
y_pred_test = vc.predict(X_test)
```

[200]:
```
print("Training Accuracy score:",accuracy_score(y_train, y_pred_train))
print("Testing Accuracy score:",accuracy_score(y_test, y_pred_test))
```

```
Training Accuracy score: 1.0
Testing Accuracy score: 0.755
```