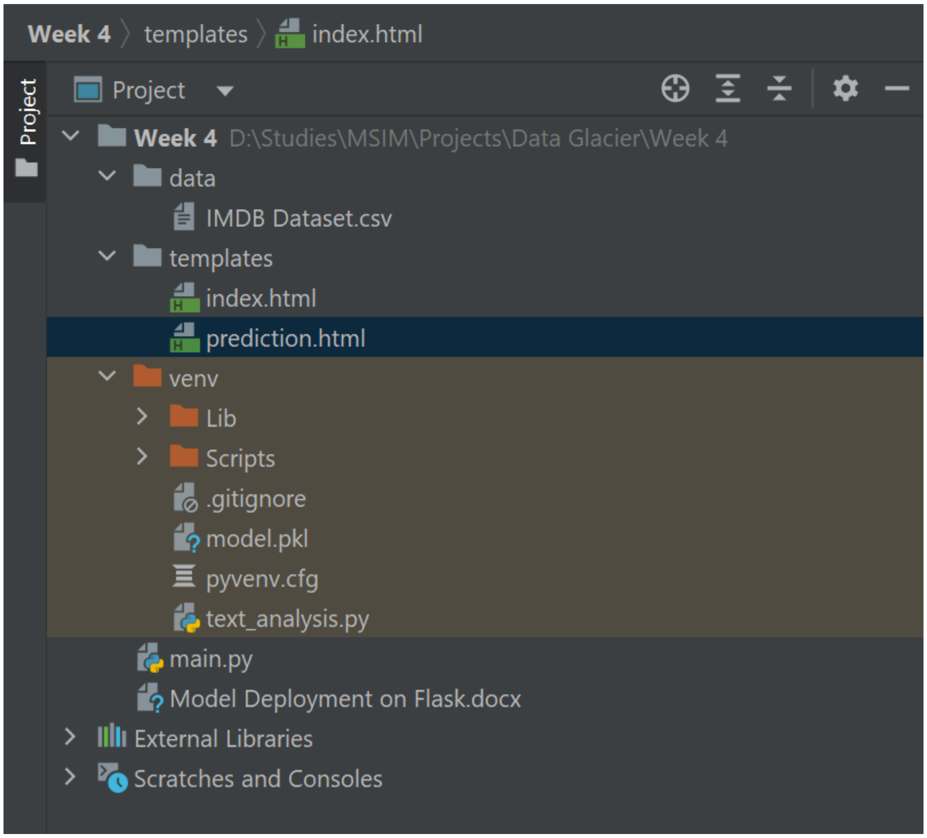
Model Deployment on Flask

Name: Mandar Parkar
Batch code: LISUM19
Submission date: 03/29/2023

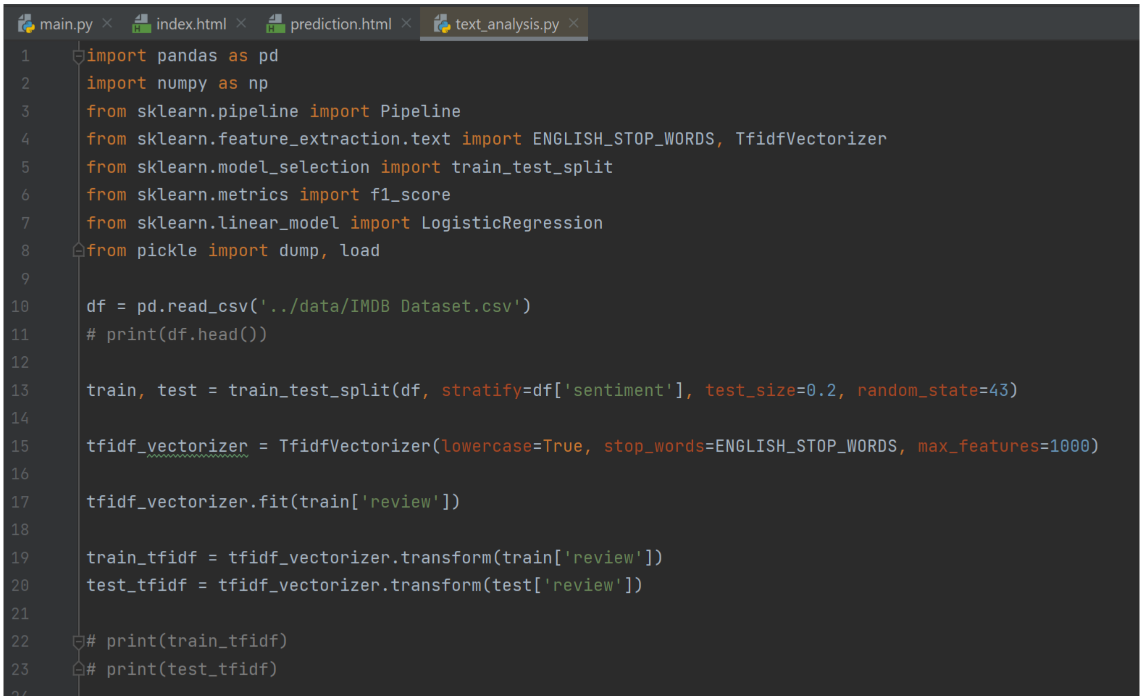
My model takes in a review from the user and determines if the sentiment of the review is positive or negative. In itself it is useless but can be scaled to take a file input from the user and output analysis of the results rather than the actual classification.

Project Structure

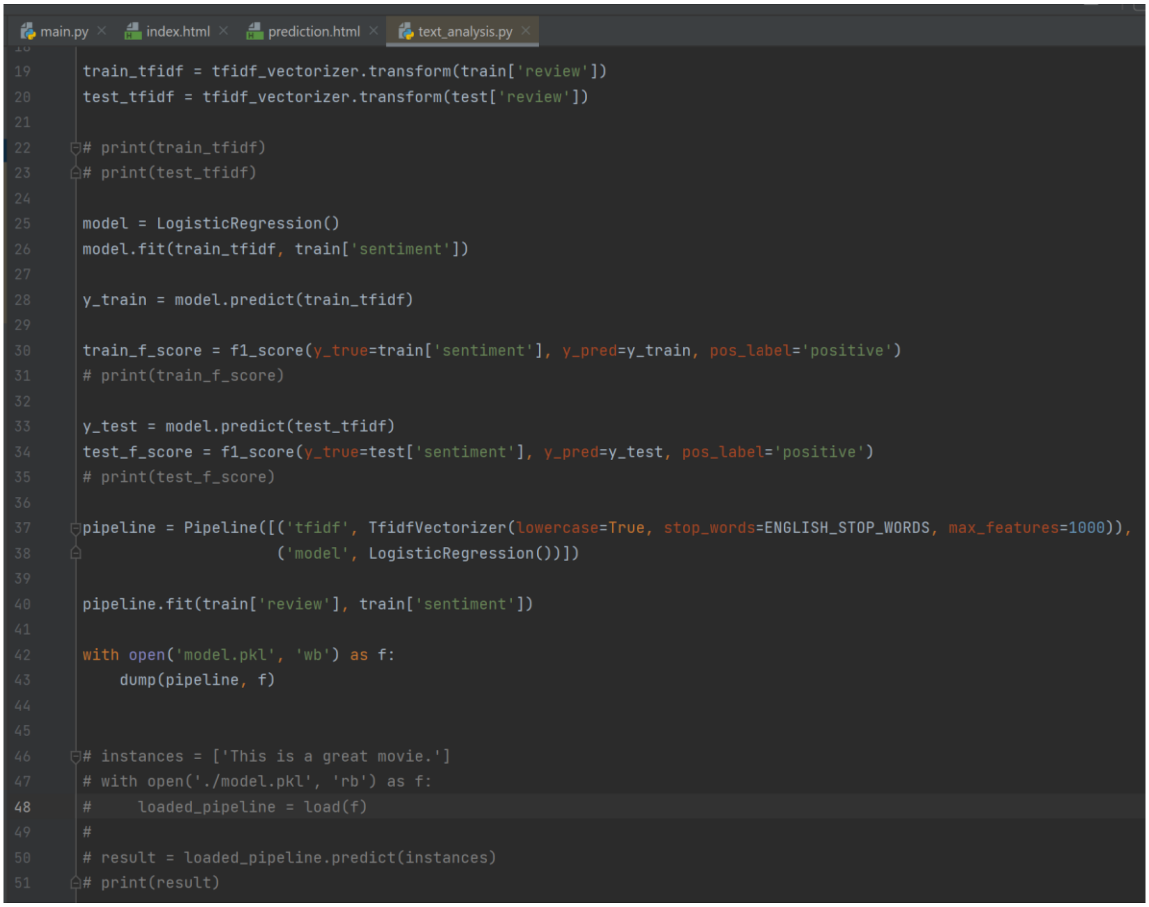


```
Week 4  D:\Studies\MSIM\Projects\Data Glacier\Week 4
  data
    IMDB Dataset.csv
  templates
    index.html
    prediction.html
  venv
    Lib
    Scripts
    .gitignore
    model.pkl
    pyvenv.cfg
    text_analysis.py
  main.py
  Model Deployment on Flask.docx
External Libraries
Scratches and Consoles
```

Model Creation

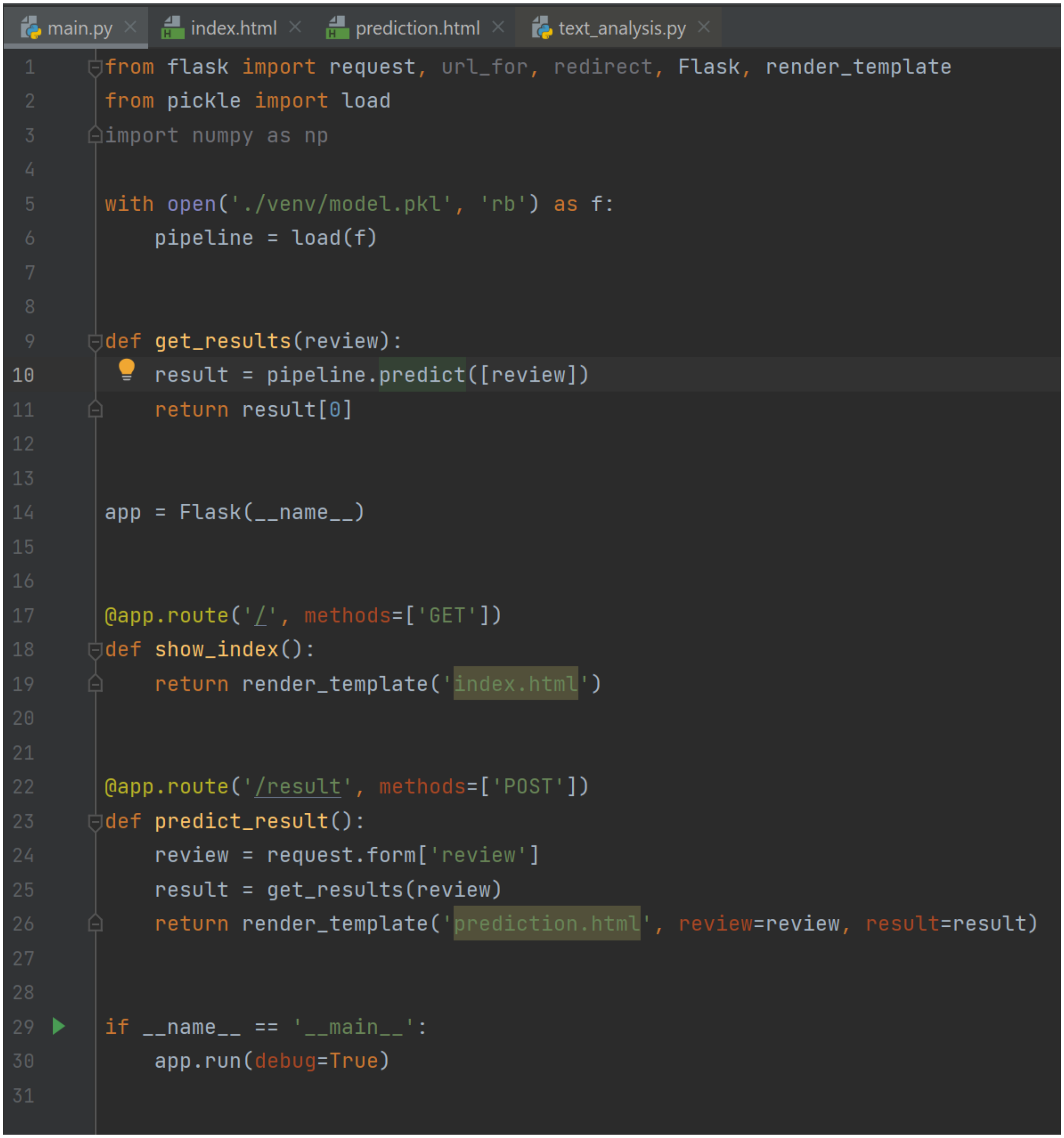


```python
import pandas as pd
import numpy as np
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import ENGLISH_STOP_WORDS, TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
from sklearn.linear_model import LogisticRegression
from pickle import dump, load

df = pd.read_csv('../data/IMDB Dataset.csv')
# print(df.head())

train, test = train_test_split(df, stratify=df['sentiment'], test_size=0.2, random_state=43)

tfidf_vectorizer = TfidfVectorizer(lowercase=True, stop_words=ENGLISH_STOP_WORDS, max_features=1000)

tfidf_vectorizer.fit(train['review'])

train_tfidf = tfidf_vectorizer.transform(train['review'])
test_tfidf = tfidf_vectorizer.transform(test['review'])

# print(train_tfidf)
# print(test_tfidf)
```



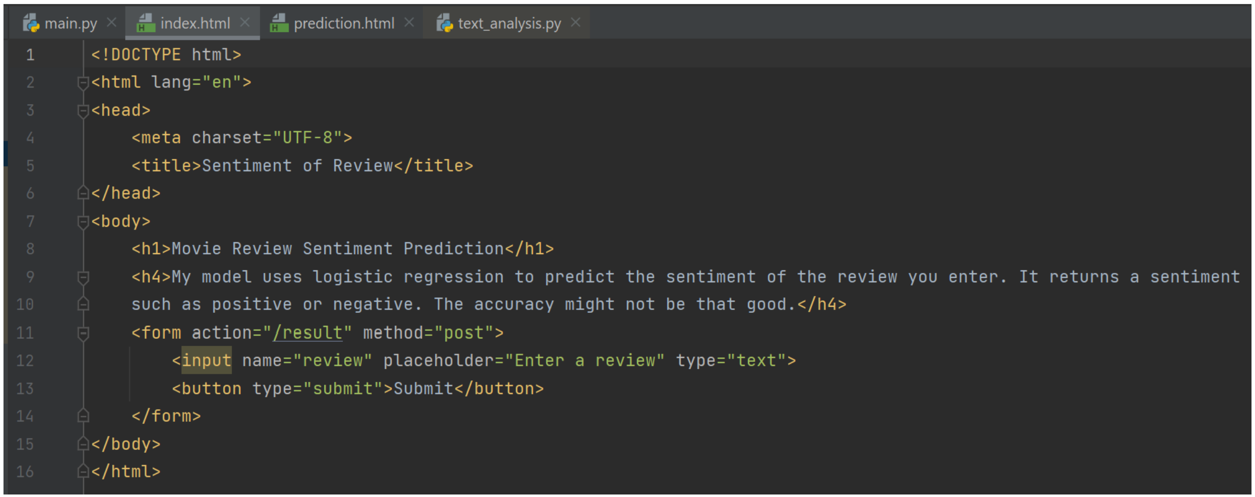
```python
train_tfidf = tfidf_vectorizer.transform(train['review'])
test_tfidf = tfidf_vectorizer.transform(test['review'])

# print(train_tfidf)
# print(test_tfidf)

model = LogisticRegression()
model.fit(train_tfidf, train['sentiment'])

y_train = model.predict(train_tfidf)

train_f_score = f1_score(y_true=train['sentiment'], y_pred=y_train, pos_label='positive')
# print(train_f_score)

y_test = model.predict(test_tfidf)
test_f_score = f1_score(y_true=test['sentiment'], y_pred=y_test, pos_label='positive')
# print(test_f_score)

pipeline = Pipeline([('tfidf', TfidfVectorizer(lowercase=True, stop_words=ENGLISH_STOP_WORDS, max_features=1000)),
                     ('model', LogisticRegression())])

pipeline.fit(train['review'], train['sentiment'])

with open('model.pkl', 'wb') as f:
    dump(pipeline, f)


# instances = ['This is a great movie.']
# with open('./model.pkl', 'rb') as f:
#     loaded_pipeline = load(f)
#
# result = loaded_pipeline.predict(instances)
# print(result)
```

Deployment

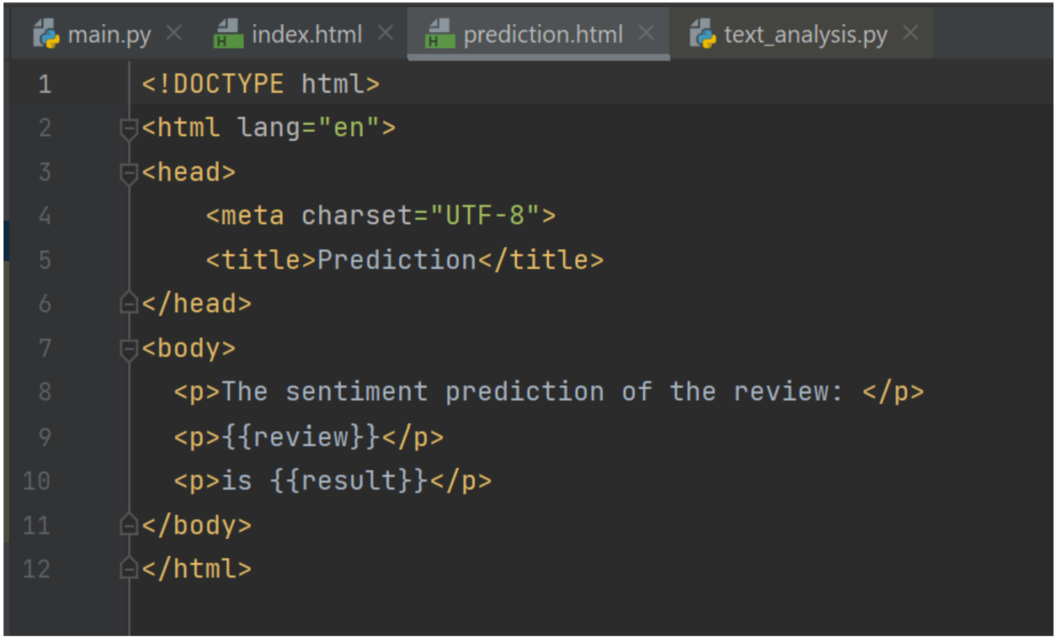


```python
from flask import request, url_for, redirect, Flask, render_template
from pickle import load
import numpy as np

with open('./venv/model.pkl', 'rb') as f:
    pipeline = load(f)


def get_results(review):
    result = pipeline.predict([review])
    return result[0]


app = Flask(__name__)


@app.route('/', methods=['GET'])
def show_index():
    return render_template('index.html')


@app.route('/result', methods=['POST'])
def predict_result():
    review = request.form['review']
    result = get_results(review)
    return render_template('prediction.html', review=review, result=result)


if __name__ == '__main__':
    app.run(debug=True)

```

Templates



```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Sentiment of Review</title>
</head>
<body>
    <h1>Movie Review Sentiment Prediction</h1>
    <h4>My model uses logistic regression to predict the sentiment of the review you enter. It returns a sentiment
    such as positive or negative. The accuracy might not be that good.</h4>
    <form action="/result" method="post">
        <input name="review" placeholder="Enter a review" type="text">
        <button type="submit">Submit</button>
    </form>
</body>
</html>
```



```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Prediction</title>
</head>
<body>
  <p>The sentiment prediction of the review: </p>
  <p>{{review}}</p>
  <p>is {{result}}</p>
</body>
</html>
```