

Heroku Deployment Tutorial

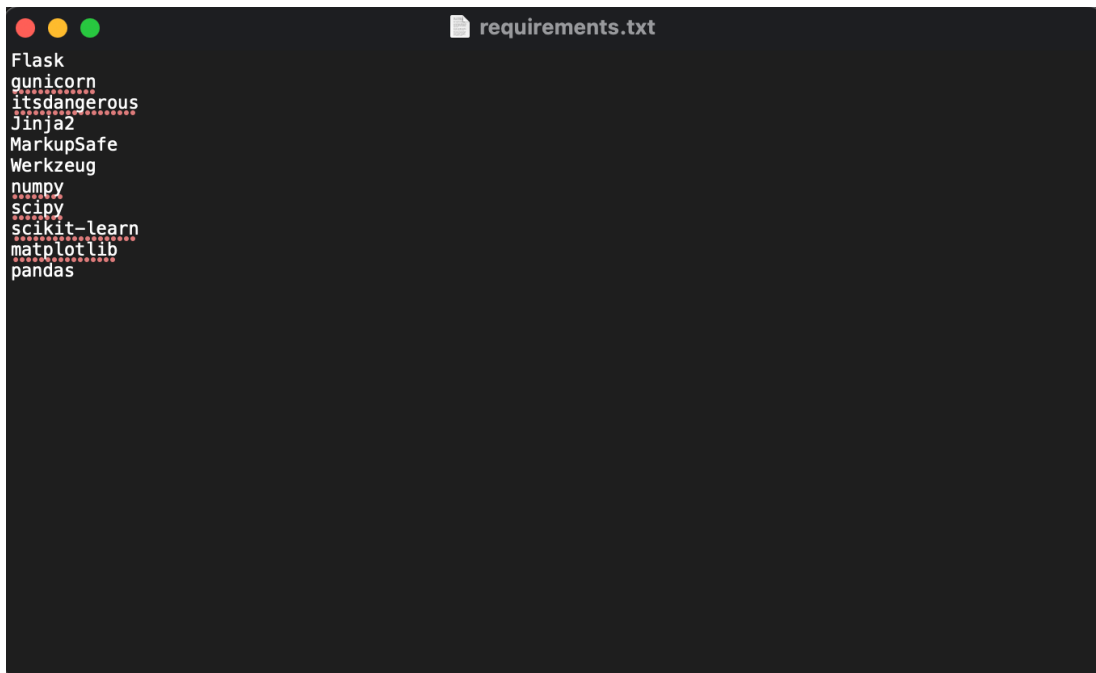
1. Create ML Model and save (pickle) it

```
model.py — Edited
Typeset LaTeX
1 import pandas as pd
2 import numpy as np
3 import sklearn
4 import pickle
5
6 df_data = pd.read_csv("adult.csv")
7
8 df_data = df_data.drop(["fnlwgt", "educational-num"], axis = 1)
9
10 col_names = df_data.columns
11 for c in col_names:
12     df_data = df_data.replace("?", np.NaN)
13     df_data = df_data.apply(lambda x: x.fillna(x.value_counts().index[0]))
14
15 category_col = ["workclass", "education", "marital-status", "occupation", "relationship",
16                "race", "gender", "native-country", "income"]
17 labelEncoder = preprocessing.LabelEncoder()
18
19 mapping_dict = {}
20 for col in category_col:
21     df_data[col] = labelEncoder.fit_transform(df_data[col])
22
23 le_name_mapping = dict(zip(labelEncoder.classes_,
24                             labelEncoder.transform(labelEncoder.classes_)))
25
26 mapping_dict[col] = le_name_mapping
27
28 X = df_data.drop("income", axis = 1)
29 Y = df_data["income"]
30
31 X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.3, random_state = 100)
32
33 rf = RandomForestClassifier(n_estimators=100, random_state=0)
34 rf.fit(X_train, y_train)
35 ypred = rf.predict(X_test)
36
37 pickle_out = open("model.pkl", "wb")
38 pickle.dump(rf, pickle_out)
39 pickle_out.close()
```

2. Create Flask files for UI and python main file (app.py) that can unpickle the machine learning model from step 1 and do predictions

```
app.py
Typeset LaTeX
1 import numpy as np
2 import pandas as pd
3 from flask import Flask, request, render_template
4 from sklearn import preprocessing
5 import pickle
6
7 app = Flask(__name__)
8 model = pickle.load(open("model.pkl", "rb"))
9 cols = ["age", "workclass", "education", "marital-status", "occupation", "relationship", "race", "gender", "capital-
10         gain", "capital-loss",
11         "hours-per-week", "native-country"]
12
13 @app.route("/")
14 def home():
15     return render_template("index.html")
16
17 @app.route("/predict", methods=["POST"])
18 def predict():
19     feature_list = request.form.to_dict()
20     feature_list = list(feature_list.values())
21     feature_list = list(map(int, feature_list))
22     final_features = np.array(feature_list).reshape(1, 12)
23
24     prediction = model.predict(final_features)
25     output = int(prediction[0])
26
27     if output == 1:
28         text = ">50K"
29     else:
30         text = "<=50K"
31
32     return render_template("index.html", prediction_text="Employee Income is {}".format(text))
33
34 if __name__ == "__main__":
35     app.run(debug=True)
```

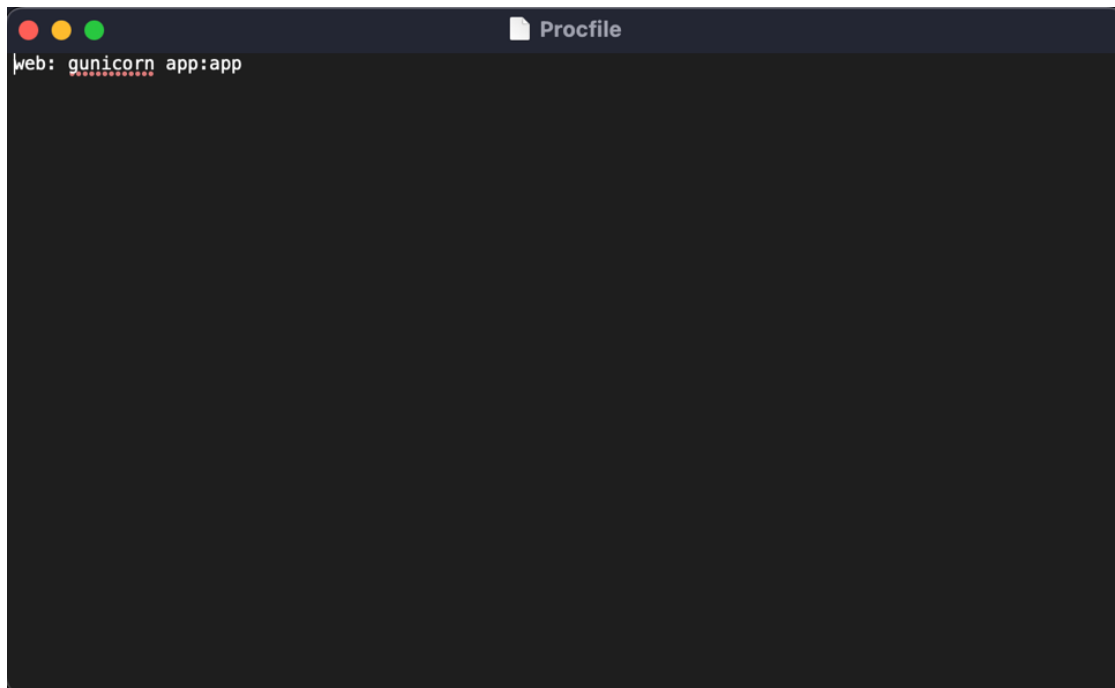
3. Create requirements.txt to setup Flask web app with all python dependencies



A screenshot of a code editor window titled "requirements.txt". The file contains a list of Python dependencies, each preceded by a red dotted line. The dependencies listed are: Flask, gunicorn, itsdangerous, Jinja2, MarkupSafe, Werkzeug, numpy, scipy, scikit-learn, matplotlib, and pandas.

```
Flask
gunicorn
itsdangerous
Jinja2
MarkupSafe
Werkzeug
numpy
scipy
scikit-learn
matplotlib
pandas
```

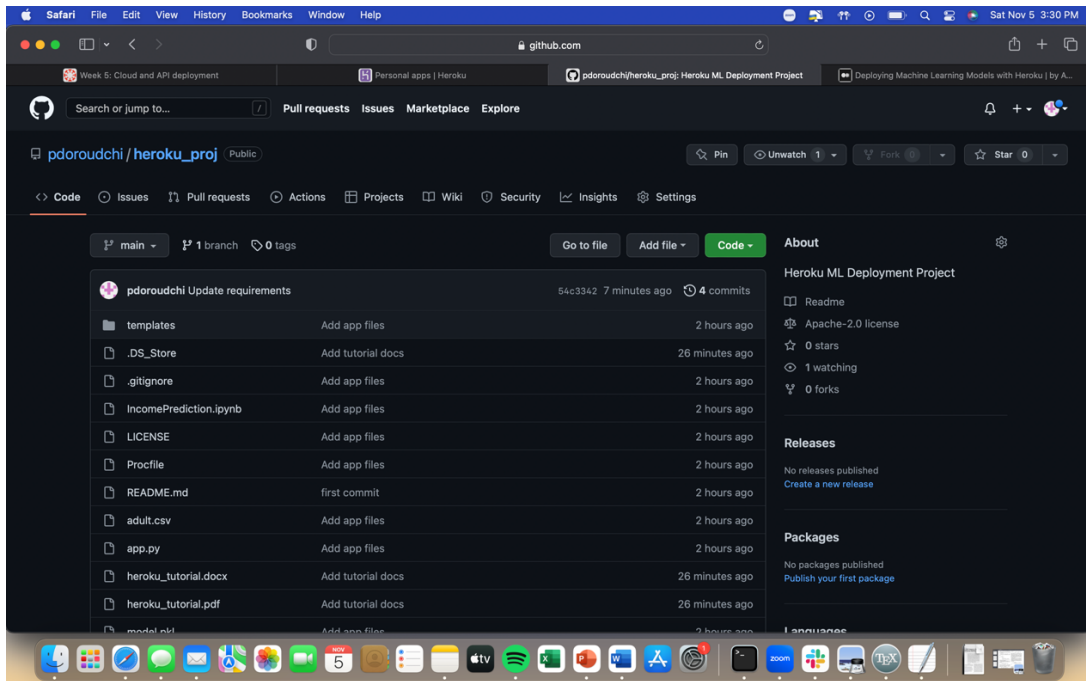
4. Create Procfile to initiate Flask app command



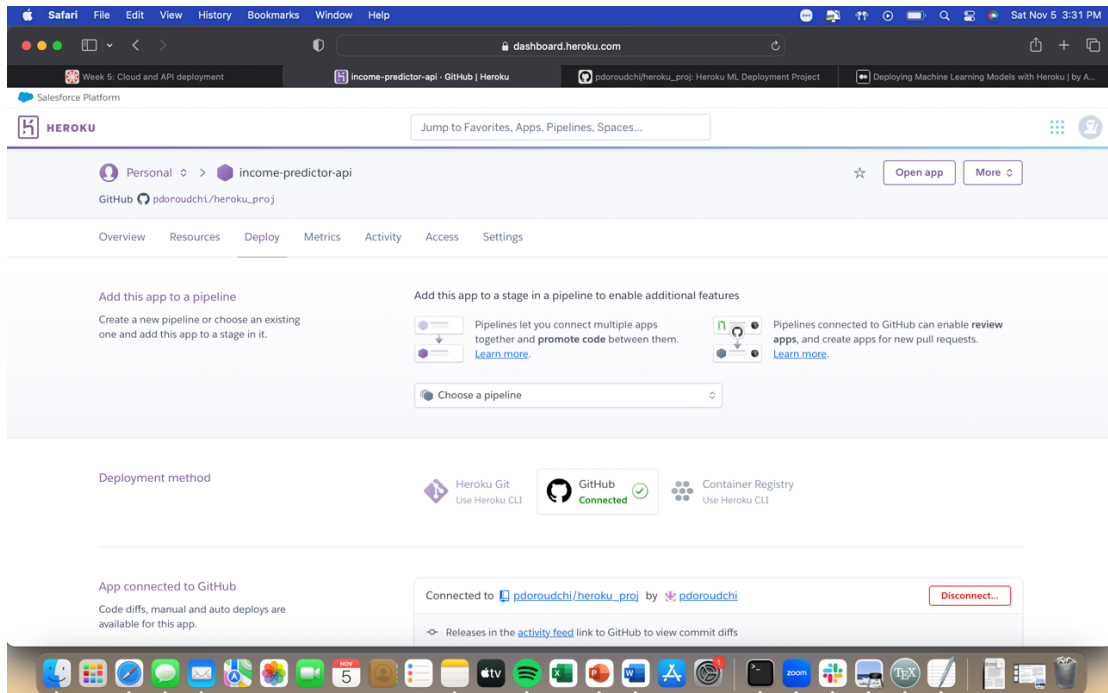
A screenshot of a code editor window titled "Procfile". The file contains a single line of text: "web: gunicorn app:app".

```
web: gunicorn app:app
```

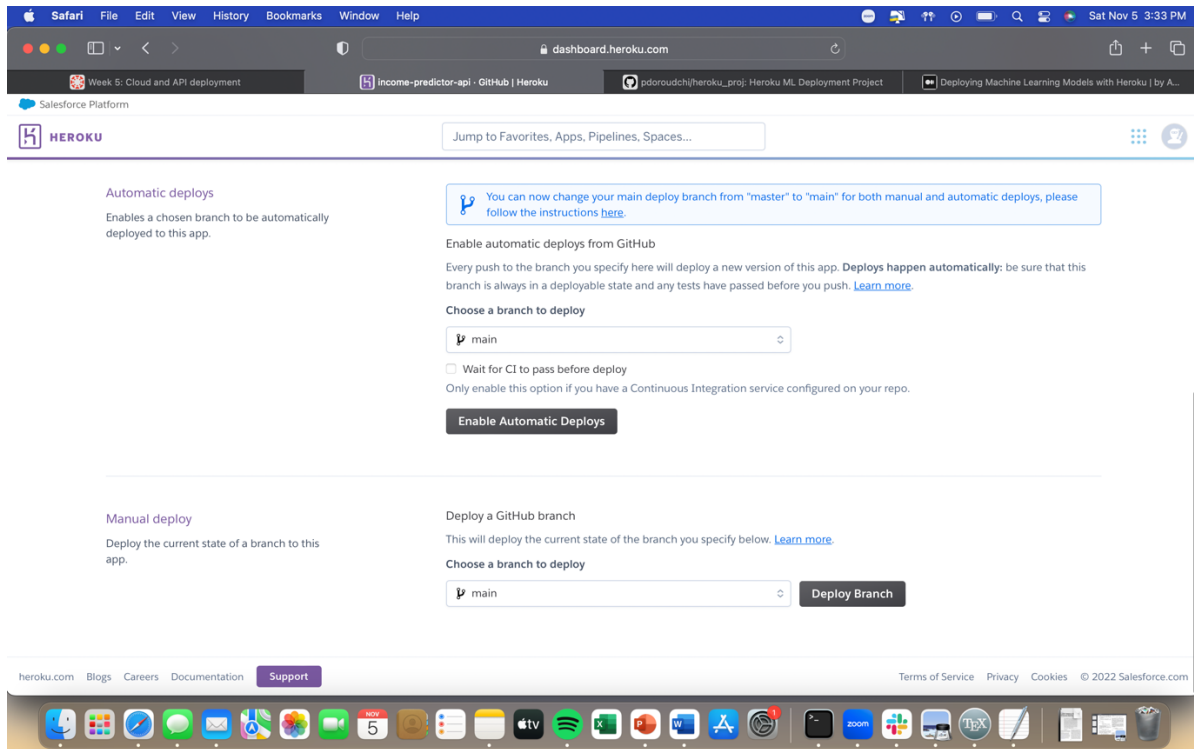
5. Commit files from Step 1, 2, 3 & 4 to GitHub repo



6. Create account/Login on Heroku, create an app, connect with GitHub repo, and select branch



7. Manually deploy on Heroku



8. Enjoy app!

