Name: Swapnil Vishwakarma

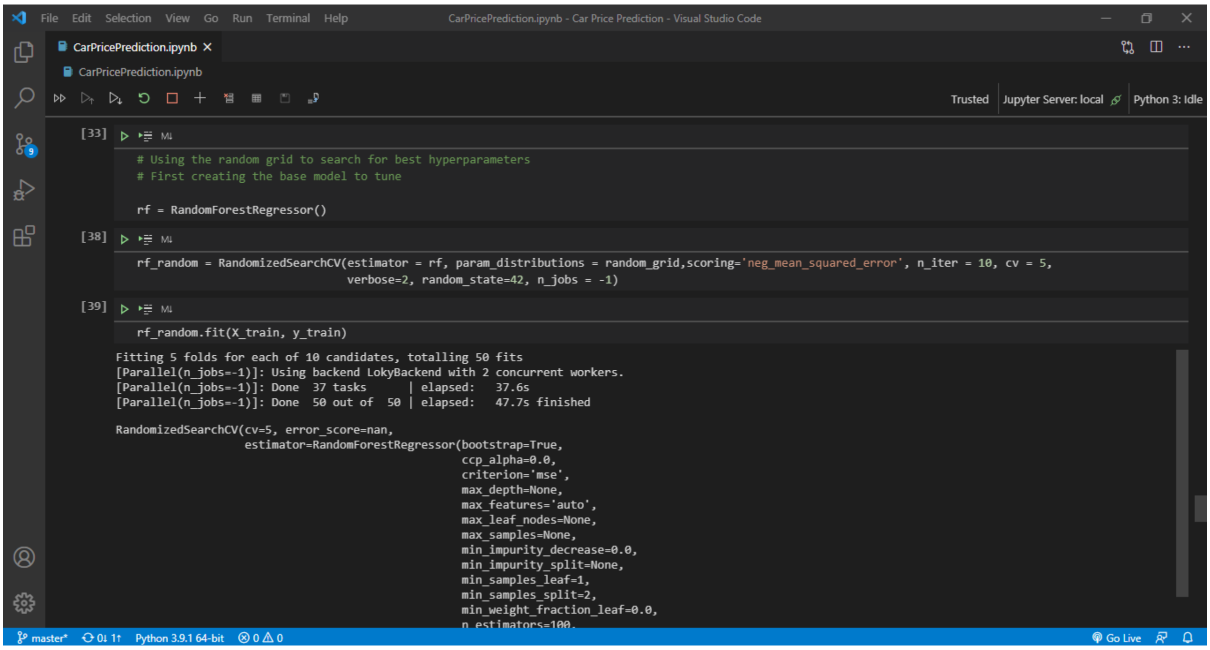
Batch Code: LISP01

Submission Date: 20-03-2021

Submitted to: Data Glacier

DEPLOYMENT PROCESS:

Step 1) Create a Machine Learning Model



[33]

```
# Using the random grid to search for best hyperparameters
# First creating the base model to tune

rf = RandomForestRegressor()
```

[38]

```
rf_random = RandomizedSearchCV(estimator = rf, param_distributions = random_grid,scoring='neg_mean_squared_error', n_iter = 10, cv = 5,
                               verbose=2, random_state=42, n_jobs = -1)
```

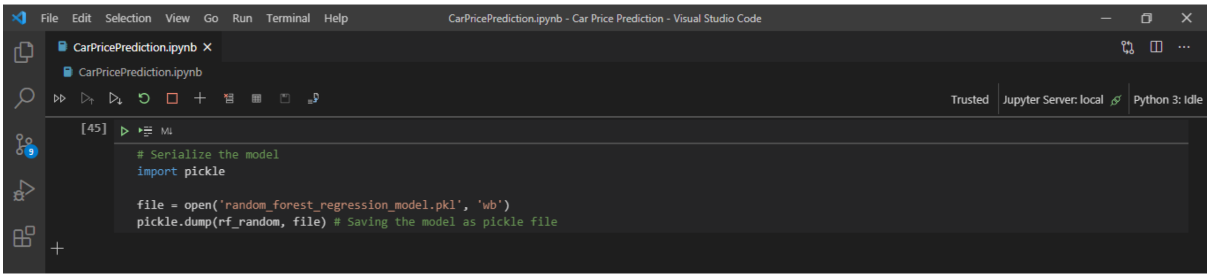
[39]

```
rf_random.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done  37 tasks      | elapsed:   37.6s
[Parallel(n_jobs=-1)]: Done  50 out of  50 | elapsed:   47.7s finished

RandomizedSearchCV(cv=5, error_score=nan,
                   estimator=RandomForestRegressor(bootstrap=True,
                                                   ccp_alpha=0.0,
                                                   criterion='mse',
                                                   max_depth=None,
                                                   max_features='auto',
                                                   max_leaf_nodes=None,
                                                   max_samples=None,
                                                   min_impurity_decrease=0.0,
                                                   min_impurity_split=None,
                                                   min_samples_leaf=1,
                                                   min_samples_split=2,
                                                   min_weight_fraction_leaf=0.0,
                                                   n_estimators=100,
```

I am using the car dataset from kaggle of cardekho.com and using Random Forest Regressor to train my model along with hyperparameter tuning.

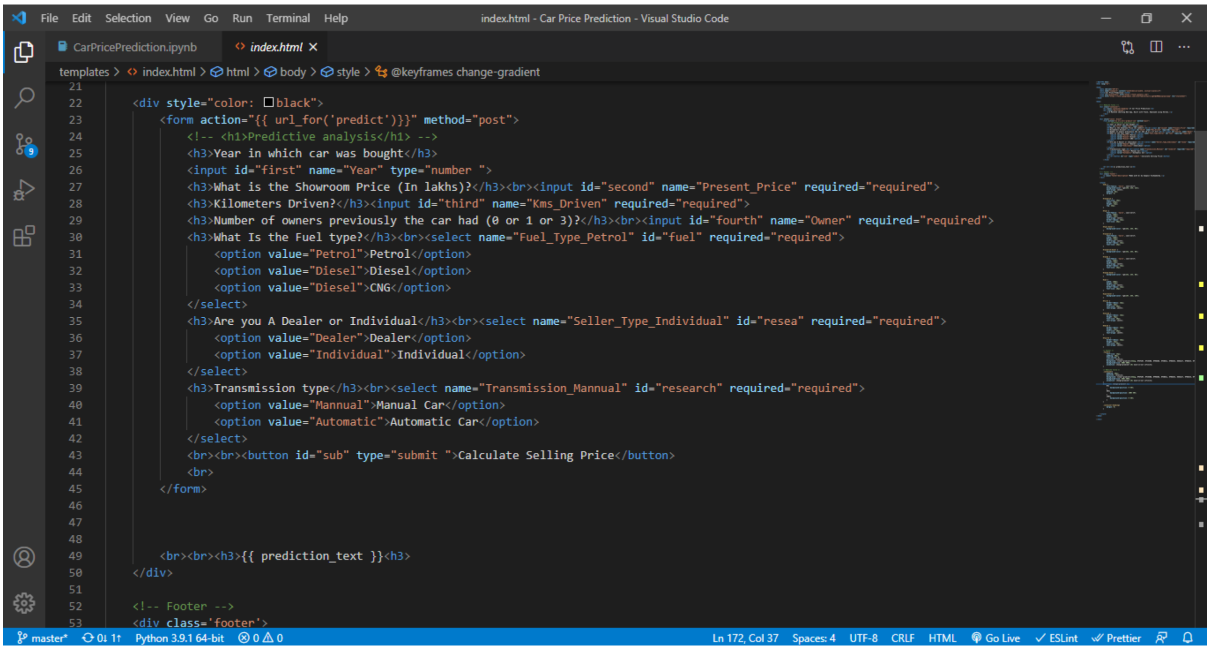
Step 2) Serialization using Pickle


File Edit Selection View Go Run Terminal Help
CarPricePrediction.ipynb - Car Price Prediction - Visual Studio Code
CarPricePrediction.ipynb
Trusted
Jupyter Server: local
Python 3: Idle
[45]

```python
# Serialize the model
import pickle

file = open('random_forest_regression_model.pkl', 'wb')
pickle.dump(rf_random, file) # Saving the model as pickle file
```

Using pickle.dump() to perform serialization using python's inbuilt module pickle.

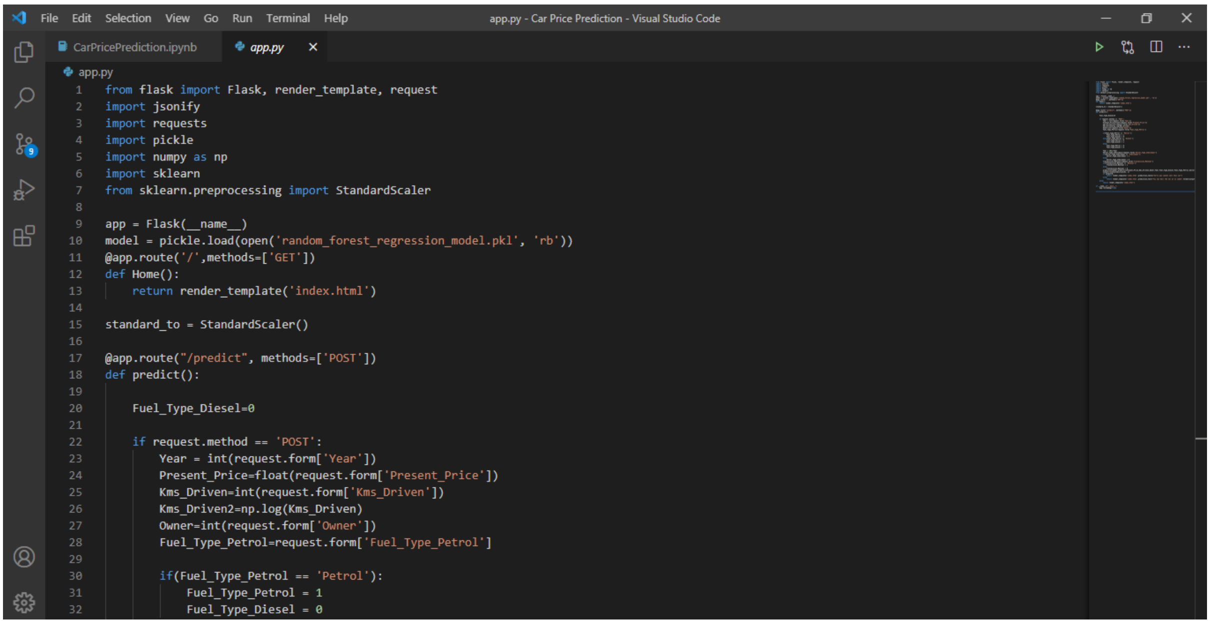
Step 3) Creating HTML Form


File Edit Selection View Go Run Terminal Help
index.html - Car Price Prediction - Visual Studio Code
CarPricePrediction.ipynb
index.html
templates > index.html > html > body > style > @keyframes change-gradient
master* 0↓ 1↑ Python 3.9.1 64-bit ⊗ 0 ⚠ 0
Ln 172, Col 37 Spaces: 4 UTF-8 CRLF HTML Go Live ✓ ESLint ✓ Prettier

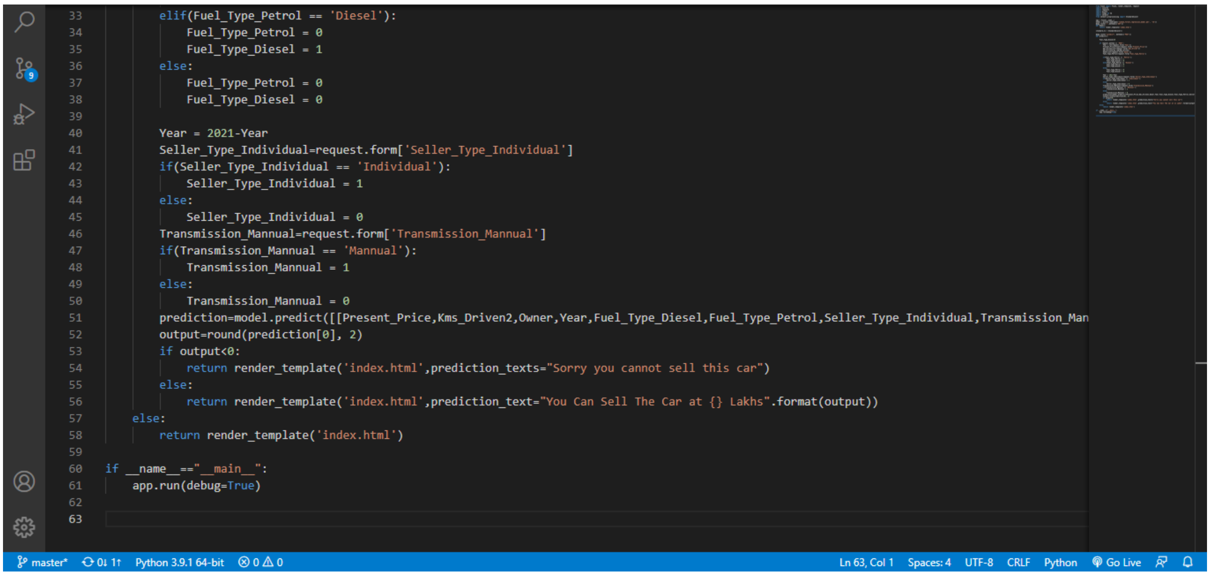
```html

<div style="color: black">
    <form action="{{ url_for('predict')}}" method="post">
        <!-- <h1>Predictive analysis</h1> -->
        <h3>Year in which car was bought</h3>
        <input id="first" name="Year" type="number ">
        <h3>What is the Showroom Price (In lakhs)?</h3><br><input id="second" name="Present_Price" required="required">
        <h3>Kilometers Driven?</h3><input id="third" name="Kms_Driven" required="required">
        <h3>Number of owners previously the car had (0 or 1 or 3)?</h3><br><input id="fourth" name="Owner" required="required">
        <h3>What Is the Fuel type?</h3><br><select name="Fuel_Type_Petrol" id="fuel" required="required">
            <option value="Petrol">Petrol</option>
            <option value="Diesel">Diesel</option>
            <option value="Diesel">CNG</option>
        </select>
        <h3>Are you A Dealer or Individual</h3><br><select name="Seller_Type_Individual" id="resea" required="required">
            <option value="Dealer">Dealer</option>
            <option value="Individual">Individual</option>
        </select>
        <h3>Transmission type</h3><br><select name="Transmission_Mannual" id="research" required="required">
            <option value="Mannual">Manual Car</option>
            <option value="Automatic">Automatic Car</option>
        </select>
        <br><br><button id="sub" type="submit ">Calculate Selling Price</button>
        <br>
    </form>



    <br><br><h3>{{ prediction_text }}<h3>
</div>

<!-- Footer -->
<div class='footer'>
```

To predict the selling price, the data is collected from new input values provided in the form and then use the model to predict the output and return the result in the form. Hence, an HTML form is used to display the result in the browser.

Step 4) Create Flask App



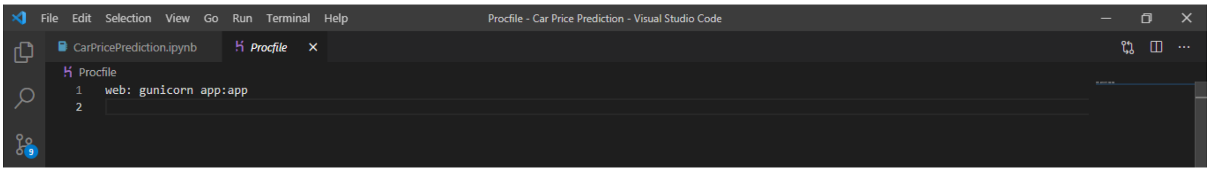
```python
from flask import Flask, render_template, request
import jsonify
import requests
import pickle
import numpy as np
import sklearn
from sklearn.preprocessing import StandardScaler

app = Flask(__name__)
model = pickle.load(open('random_forest_regression_model.pkl', 'rb'))
@app.route('/',methods=['GET'])
def Home():
    return render_template('index.html')

standard_to = StandardScaler()

@app.route("/predict", methods=['POST'])
def predict():

    Fuel_Type_Diesel=0

    if request.method == 'POST':
        Year = int(request.form['Year'])
        Present_Price=float(request.form['Present_Price'])
        Kms_Driven=int(request.form['Kms_Driven'])
        Kms_Driven2=np.log(Kms_Driven)
        Owner=int(request.form['Owner'])
        Fuel_Type_Petrol=request.form['Fuel_Type_Petrol']

        if(Fuel_Type_Petrol == 'Petrol'):
            Fuel_Type_Petrol = 1
            Fuel_Type_Diesel = 0
```



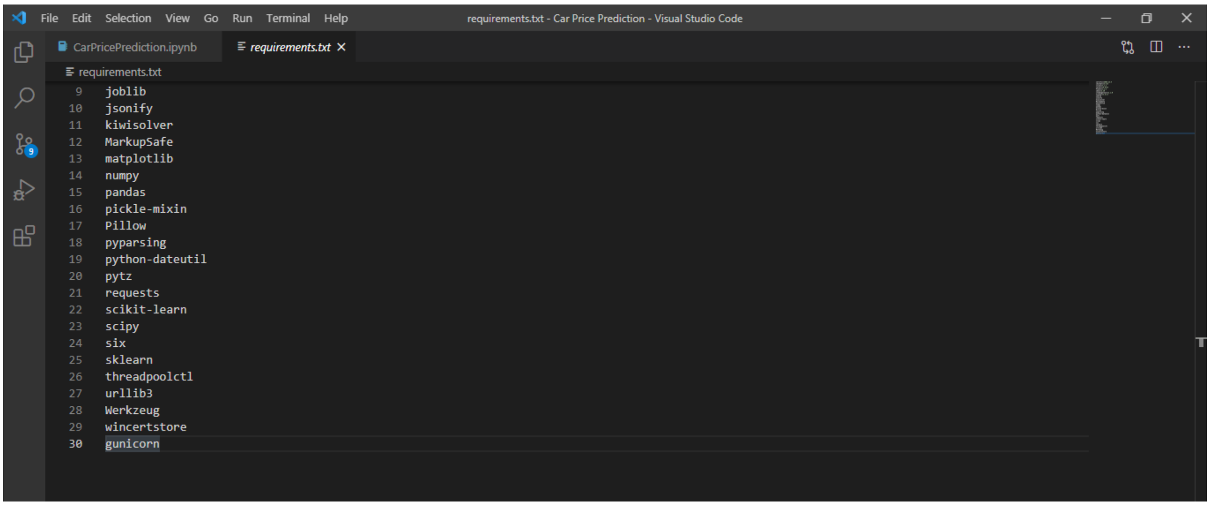
```python
        elif(Fuel_Type_Petrol == 'Diesel'):
            Fuel_Type_Petrol = 0
            Fuel_Type_Diesel = 1
        else:
            Fuel_Type_Petrol = 0
            Fuel_Type_Diesel = 0

        Year = 2021-Year
        Seller_Type_Individual=request.form['Seller_Type_Individual']
        if(Seller_Type_Individual == 'Individual'):
            Seller_Type_Individual = 1
        else:
            Seller_Type_Individual = 0
        Transmission_Mannual=request.form['Transmission_Mannual']
        if(Transmission_Mannual == 'Mannual'):
            Transmission_Mannual = 1
        else:
            Transmission_Mannual = 0
        prediction=model.predict([[Present_Price,Kms_Driven2,Owner,Year,Fuel_Type_Diesel,Fuel_Type_Petrol,Seller_Type_Individual,Transmission_Man
        output=round(prediction[0], 2)
        if output<0:
            return render_template('index.html',prediction_texts="Sorry you cannot sell this car")
        else:
            return render_template('index.html',prediction_text="You Can Sell The Car at {} Lakhs".format(output))
    else:
        return render_template('index.html')

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

```

To host the HTML form, a Flask web app is created where the pickle file is read using pickle.load(). A predict() fuction is created which takes the input from homepage (HTML homepage), the model will predict the selling price return the result.

Step 5) Create configuration files


File Edit Selection View Go Run Terminal Help
Procfile - Car Price Prediction - Visual Studio Code
CarPricePrediction.ipynb
Procfile

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


File Edit Selection View Go Run Terminal Help
requirements.txt - Car Price Prediction - Visual Studio Code
CarPricePrediction.ipynb
requirements.txt

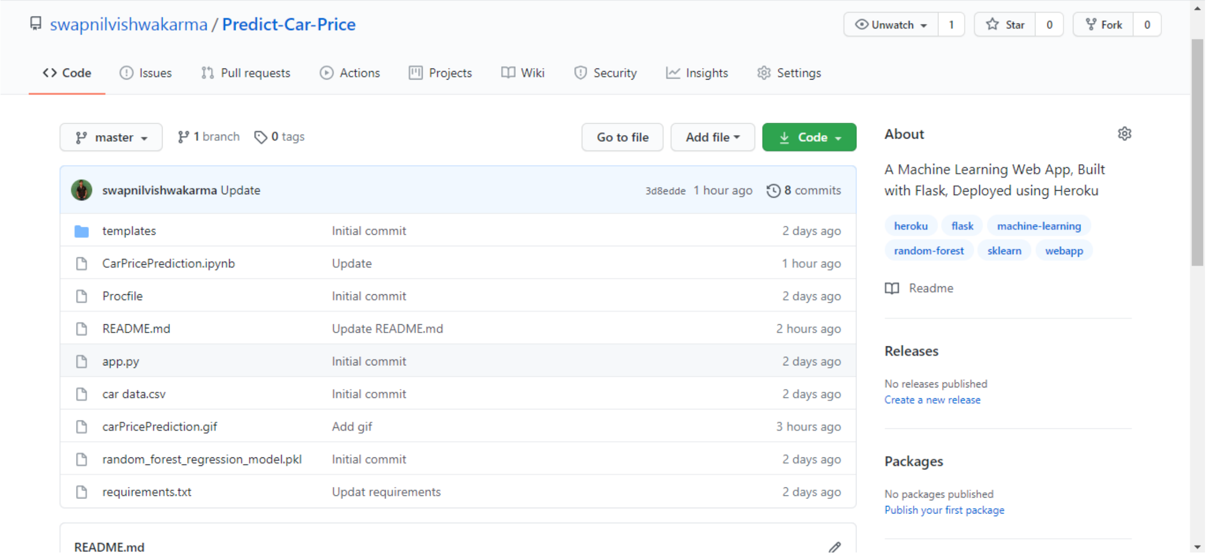
```
joblib
jsonify
kiwisolver
MarkupSafe
matplotlib
numpy
pandas
pickle-mixin
Pillow
pyparsing
python-dateutil
pytz
requests
scikit-learn
scipy
six
sklearn
threadpoolctl
urllib3
Werkzeug
wincertstore
gunicorn
```

For deployment, Procfile and requirements.txt files are created.

Procfile uses Gunicorn which is a pure-Python HTTP server for WSGI applications and it acts as a liaison in between the web application and the webserver.

Requirements.txt file contains all libraries and their dependencies.

Step 6) Commit files in Github Repo



swapnilvishwakarma Update 3d8edde 1 hour ago 8 commits

templates	Initial commit	2 days ago
CarPricePrediction.ipynb	Update	1 hour ago
Procfile	Initial commit	2 days ago
README.md	Update README.md	2 hours ago
app.py	Initial commit	2 days ago
car data.csv	Initial commit	2 days ago
carPricePrediction.gif	Add gif	3 hours ago
random_forest_regression_model.pkl	Initial commit	2 days ago
requirements.txt	Updat requirements	2 days ago

README.md

About

A Machine Learning Web App, Built with Flask, Deployed using Heroku

heroku flask machine-learning random-forest sklearn webapp

Readme

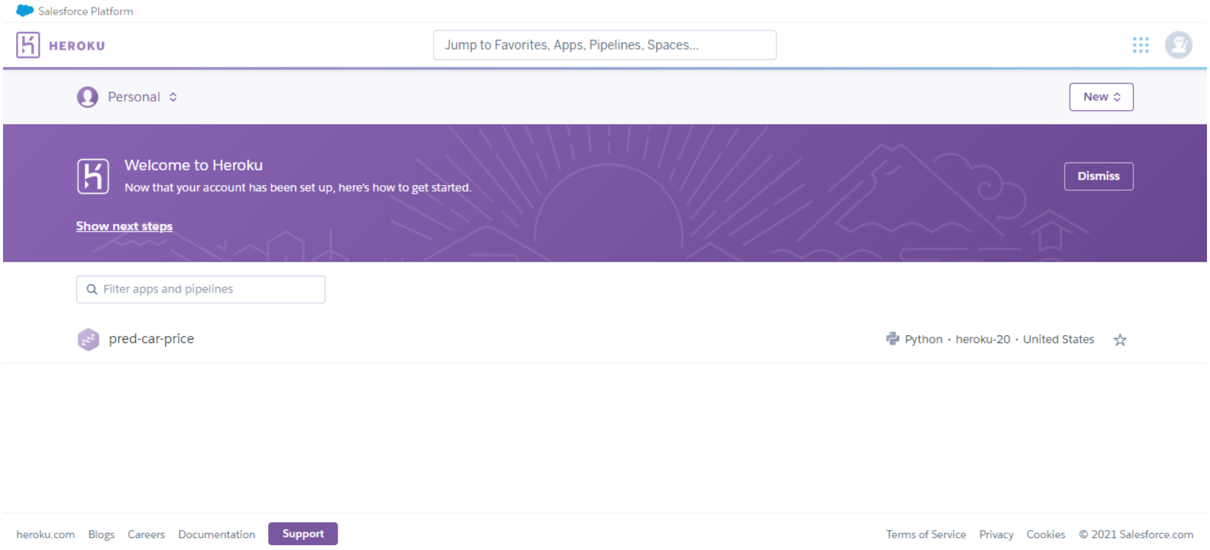
Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Step 7) Link Github Repo to Heroku and Deploy



Welcome to Heroku

Now that your account has been set up, here's how to get started.

Show next steps

pred-car-price Python · heroku-20 · United States

After creating a free account on Heroku, connect it to your Github.

To deploy a new app, click on create new app and connect to the Github repo which you want to deploy and click deploy branch. Now the web app is ready publically available.