

# Deployment on Flask

**Name:** Ammar Sidhu

**Batch Code:** LISUM14

**Submission Date:** 10/28/2022

**Submitted To:** Data Glacier on Canvas

**Email:** ammarsidhu@outlook.com

## Deployment Steps

### Step 1: Model Building

Created *Linear Regression Model* that predicts house prices in the City of Hamilton, Ontario (Canada) with Census Tract data.

#### 1. Preparing the Tools

```
In [1]: # Importing EDA and Plotting Libraries
import pandas as pd
import numpy as np
import geopandas as gpd

# Regression Model for Machine Learning
from sklearn.linear_model import LinearRegression

# Regression Model Tools
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.pipeline import Pipeline

# Model Deployment Tools
from flask import Flask, request, jsonify, render_template
import json
import pickle

# Ignore warnings
import warnings
warnings.filterwarnings("ignore")
```

## 2. Importing Data

```
In [2]: # Hamilton House Pricing & Boundary Data
url = "https://raw.githubusercontent.com/gisUTM/GGR376/master/Lab_1/houseValues.geojson"
house_prices_boundaries = gpd.read_file(url)
house_prices_boundaries.head(5)
```

```
Out[2]:
```

	CTUID	houseValue	geometry
0	5370001.00	420276.0	POLYGON ((-79.85588 43.18790, -79.85592 43.187...
1	5370120.02	601551.0	POLYGON ((-79.84582 43.16920, -79.84637 43.167...
2	5370140.03	525073.0	POLYGON ((-79.89977 43.33088, -79.89977 43.330...
3	5370140.04	524777.0	POLYGON ((-79.89288 43.32009, -79.89228 43.328...
4	5370001.08	400617.0	POLYGON ((-79.85382 43.19320, -79.85380 43.192...

```
In [3]: # Hamilton Housing Features Data
house_attributes = pd.read_csv("hamilton_census_data.csv")
house_attributes.head(5)
```

```
Out[3]:
```

	CTUID	priv_dwellings_by_bedroom	priv_dwellings_by_rooms	major_repairs	monthly_housing_costs	percent_mortgage	income_after_tax	house_by_pers
0	5370000.00	293345	293345	18370	1294	60.6	66100	
1	5370001.01	780	780	35	1269	57.9	66530	
2	5370001.02	1780	1780	70	1493	70.2	72590	
3	5370001.04	1900	1895	50	1387	59.7	83257	
4	5370001.05	1805	1810	45	1569	65.3	75136	

```
In [4]: # Merge Features Data & House Pricing/Boundary Data
df = house_prices_boundaries.merge(house_attributes, on = "CTUID")
df.head(3)
```

```
Out[4]:
```

	CTUID	houseValue	geometry	priv_dwellings_by_bedroom	priv_dwellings_by_rooms	major_repairs	monthly_housing_costs	percent_mortgage	income
0	5370001.00	420276.0	POLYGON ((-79.85588 43.18790, -79.85592 43.187...	1390	1390	25	1612	72.3	
1	5370120.02	601551.0	POLYGON ((-79.84582 43.16920, -79.84637 43.167...	890	885	75	1480	55.5	
2	5370140.03	525073.0	POLYGON ((-79.89977 43.33088, -79.89977 43.330...	2505	2505	50	1883	78.3	

## 3. Data Preprocessing

```
In [6]: # Split the data into X and y
X = df.drop("houseValue", axis = 1)
y = df["houseValue"]
```

```
In [7]: # Inspect X
X.head(5)
```

```
Out[7]:
```

	priv_dwellings_by_bedroom	priv_dwellings_by_rooms	major_repairs	monthly_housing_costs	percent_mortgage	income_after_tax	house_by_person_per_room
0	1390	1390	25	1612	72.3	78976	1390
1	890	885	75	1480	55.5	87211	885
2	2505	2505	50	1883	78.3	98824	2505
3	1320	1315	25	1705	63.5	94948	1320
4	2050	2050	35	1513	66.7	80575	2050

```
In [8]: # Inspect y
y.head(5)
```

```
Out[8]:
```

0	420276.0
1	601551.0
2	525073.0
3	524777.0
4	400617.0

Name: houseValue, dtype: float64

```
In [9]: # Set random seed for consistency and reproducibility
np.random.seed(42)

# Split data into train & sets
X_train, X_test, y_train, y_test = train_test_split(X, # independent variables
                                                    y, # dependent variable
                                                    test_size = 0.2) # percentage of data to use for test set

# Sizes of test & training sets
# Sizes of test & training sets
print("The shape of X_train is:", X_train.shape)
print("The shape of X_test is: ", X_test.shape)
print("The shape of y_train is:", y_train.shape)
print("The shape of y_test is: ", y_test.shape)

The shape of X_train is: (149, 10)
The shape of X_test is: (38, 10)
The shape of y_train is: (149,)
The shape of y_test is: (38,)
```

## 4. Model Building

```
In [10]: # Instantiate Linear Regression Model
lm_reg = LinearRegression()

# Fit Model to Training Data
lm_reg.fit(X_train, y_train)

Out[10]: LinearRegression()

In [11]: # Obtain Predictions of Model from Training Data
y_pred = lm_reg.predict(X_test)
print(y_pred)

[596387.12803    480896.92267824  476224.5802106   367575.11673828
 351865.10308534  456056.27568041  192322.88930761  259228.3522551
 375951.18284039  452391.01240462  408266.73433746  284941.20311671
 506087.78903799  342262.56313031  419726.11973444  384849.97199765
 444869.90434751  478481.94049903  196444.74973795  283961.41623627
 489293.36258692  430700.39853896  601440.69392218  380113.38579982
 304920.81985184  850140.29680267   13538.43240843  403399.53371897
 344473.79676943  478111.17279516  375967.31249919  699060.43153693
 297323.75558971  464661.57886319  368901.60541925  273769.63600779
 556244.11125234  496297.49030709]
```

## Step 2: Saving Model

Saved trained *Linear Regression Model* to local device using *Pickle* and re-imported model to test predictions with mock-data.

## 5. Saving the Model

```
In [12]: # Save Model as Pickle File to Local Device
pickle.dump(lm_reg, open('model.pkl', 'wb'))

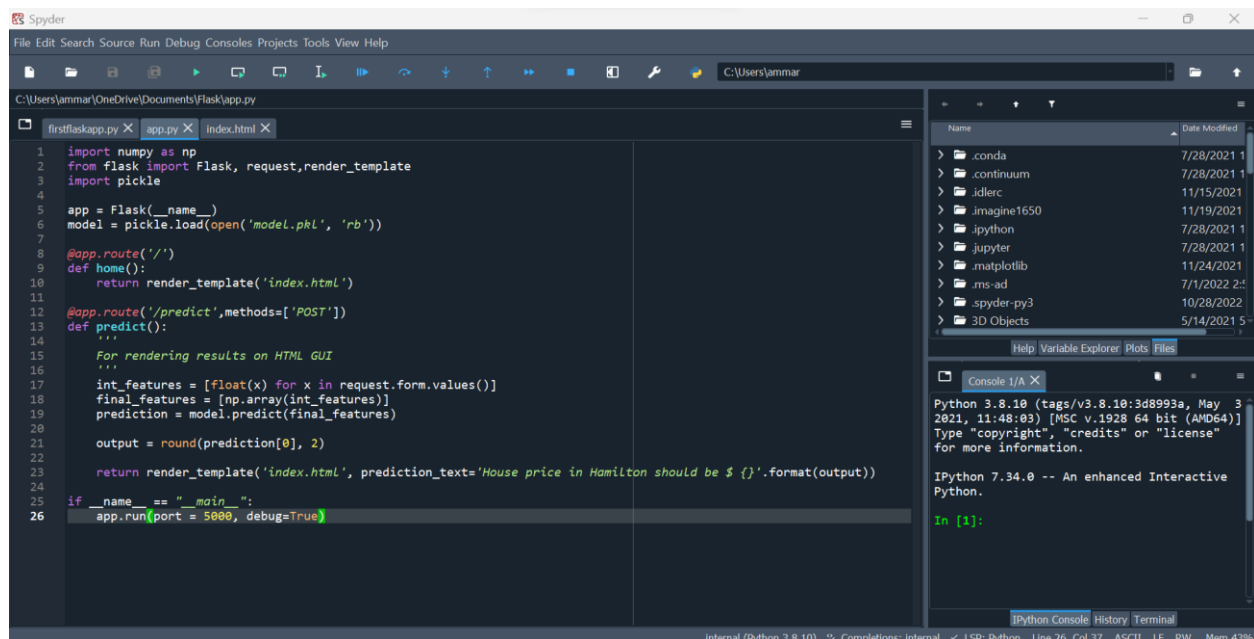
In [13]: # Load Linear Regression Model
lm_model = pickle.load(open('model.pkl', 'rb'))

# Test Model Predictions with Dummy Data
print(lm_model.predict([[1390, 1390, 25, 1612, 72.3, 78796, 1395, 3.3, 4566, 1.97]]))

[439013.16362958]
```

## Step 3: Model Deployment with Flask

Created a *Flask script* to deploy the model by allowing users to input Hamilton house features and submit their input by pressing the '*Predict*' button to receive a prediction of the house's price.

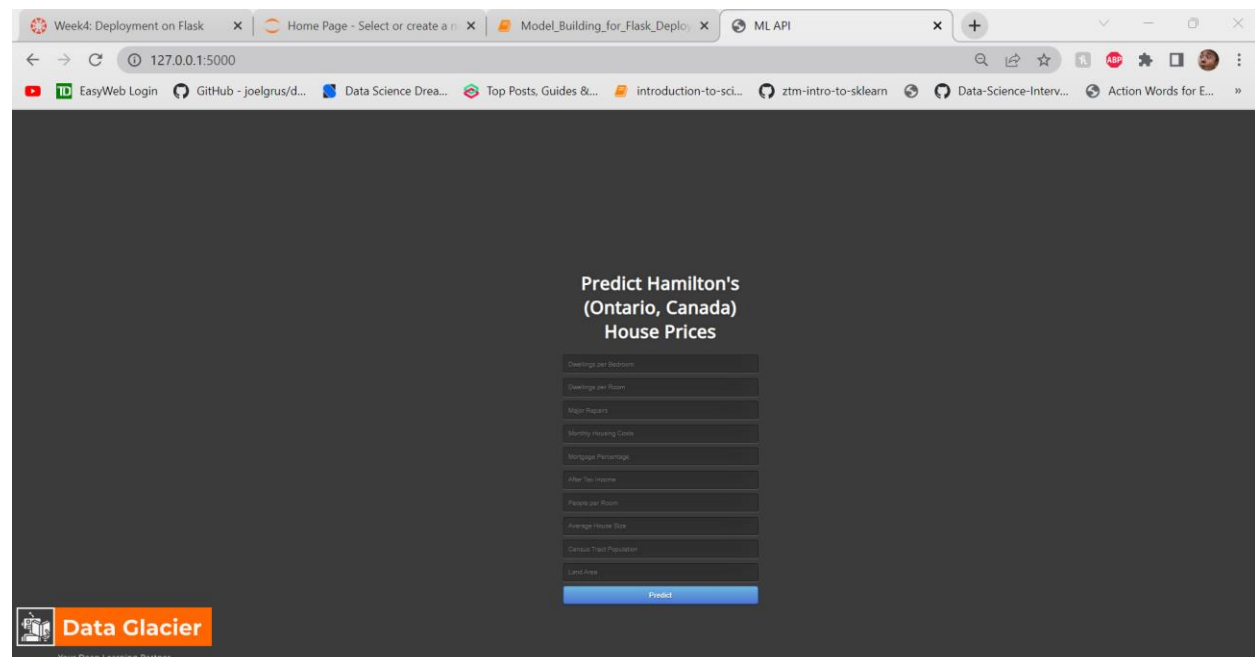


## Step 4: Acquiring Flask App's URL from CMD

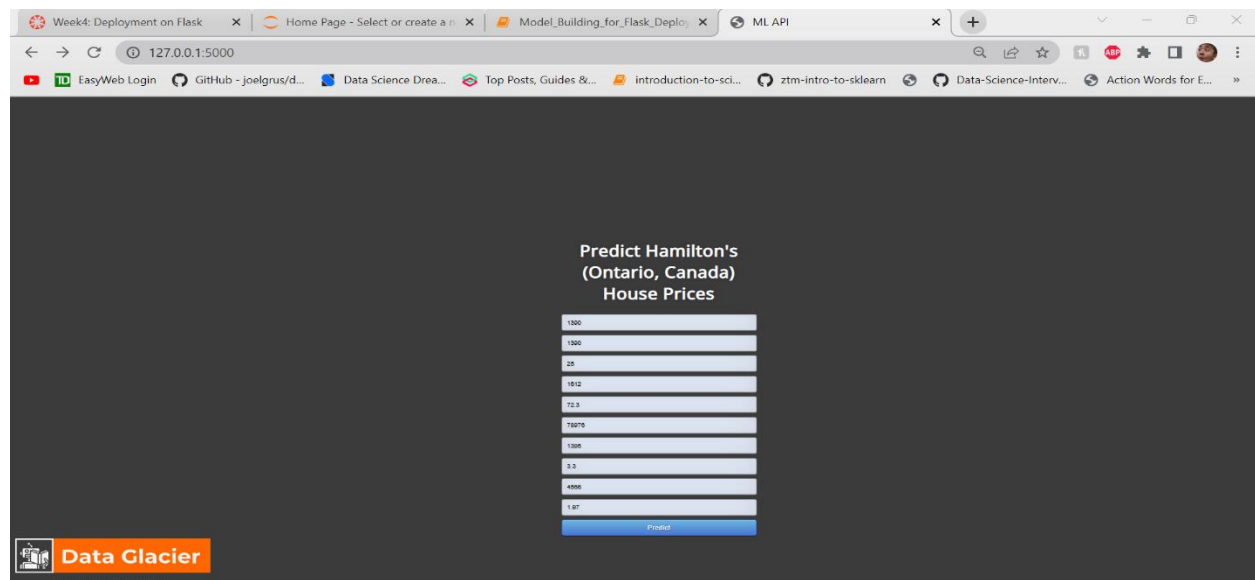
```
C:\Windows\System32\cmd.exe - python app.py
Microsoft Windows [Version 10.0.22621.674]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ammar\OneDrive\Documents\Flask>python app.py
C:\Users\ammar\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\base.py:329: UserWarning: Trying to unpickle estimator LinearRegression from version 1.0.1
when using version 1.1.3. This might lead to breaking code or invalid results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
C:\Users\ammar\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\base.py:329: UserWarning: Trying to unpickle estimator LinearRegression from version 1.0.1
when using version 1.1.3. This might lead to breaking code or invalid results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
* Debugger is active!
* Debugger PIN: 388-856-751
```

## Step 5: Inputting URL to Create Flask Web App



## Step 6: Inputting House Features for Predictions



## Step 7: Acquire Predictions by Pressing Prediction Button

Week4: Deployment on Flask x Home Page - Select or create a n x Model\_Building\_for\_Flask\_Deploy x ML API

127.0.0.1:5000/predict

EasyWeb Login GitHub - joelgrus/d... Data Science Drea... Top Posts, Guides &... introduction-to-sci... ztm-intro-to-sklearn Data-Science-Interv... Action Words for E... »

### Predict Hamilton's (Ontario, Canada) House Prices

- Overings per Bedroom
- Overings per Room
- Water Features
- Monthly Heating Costs
- Mortgage Percentage
- After Tax Income
- Prices per Room
- Average House Size
- Current Total Population
- Land Price

Predict

House price in Hamilton should be \$  
439684.46

**Data Glacier**  
Your Deep Learning Partner