

# **Data Science Internship at Data Glacier**

# **Week 4: Deployment on Flask**

Name: Chooladeva Piyasiri

Batch Code: LISUM08

Date: 28 February 2023

Submitted to: Data Glacier

1. Project Introduction

Flask is a web application framework written in Python. It has multiple modules that make it

easier for a web developer to write applications without having to worry about the details like

protocol management, thread management, etc. Flask gives is a variety of choices for developing

web applications and it gives us the necessary tools and libraries that allow us to build a web

application.

Model deployment is the process of putting machine learning models into production. This

makes the model's predictions available to users, developers or systems, so they can make

business decisions based on data, interact with their application

**Model**: Predicting house prices in certain metropolitan areas based in Malborne, Australia.

2. Dataset Description

I have taken data from Kaggle. I am going to use this data for the model. The full data set contains

18396 rows of data with 21 attributes. Notes on Specific Variables are given below.

**Rooms**: Number of rooms

> Price: Price in dollars

> **Suburb**: The residential area on the outskirts of a city or large town.

➤ Method:

• S - property sold;

• SP - property sold prior;

PI - property passed in;

PN - sold prior not disclosed;

SN - sold not disclosed;

NB - no bid;

VB - vendor bid;

- W withdrawn prior to auction;
- SA sold after auction;
- SS sold after auction price not disclosed.
- N/A price or highest bid not available.

#### > Type:

- br bedroom(s);
- h house, cottage, villa, semi, terrace;
- u unit, duplex;
- t townhouse;
- dev site development site;
- res other residential.
- > SellerG: Real Estate Agent
- > Date: Date sold
- > **Distance**: Distance from CBD
- **Regionname**: General Region (West, North West, North, North east ...etc.)
- **Propertycount**: Number of properties that exist in the suburb.
- **Bedroom2**: Scraped # of Bedrooms (from different source)
- > Bathroom: Number of Bathrooms
- > Car: Number of carspots
- > Landsize: Land Size
- > BuildingArea: Building Size
- > CouncilArea: Governing council for the area
- > Latitude: Latitude
- **Longitude**: Longitude
- > YearBuilt: Year the house was built
- > Address: Address of the property
- **Postcode**: Postcode of the property

### 3. Model Building

```
import pandas as pd
import numpy as np
from sklearn.model selection import train test split
from sklearn.ensemble import RandomForestRegressor
from sklearn import metrics
from sklearn.model_selection import RandomizedSearchCV
import pickle
import warnings
warnings.filterwarnings('ignore')
data=pd.read_csv('melb_data.csv',index_col=0)
data=data.dropna(axis=0)
y=data["Price"]
data_features=['Rooms','Bedroom2','Bathroom','Car','Landsize','BuildingArea','Lattitude','Longtitude']
X=data[data_features]
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.2, random_state=1)
rf model = RandomForestRegressor(random_state=1)
rf_model.fit(X_train, y_train)
y_pred = rf_model.predict(X_test)
# Evaluating the Model
#print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
#print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
#score= rf_model.score(X_test,y_test) #R2 Score
#print(score)
```

#### **Hyperparameter Tuning**

```
# Number of trees in random forest
n_{estimators} = [int(x) \text{ for } x \text{ in } np.linspace(start = 200, stop = 2000, num = 10)]
# Number of features to consider at every split
max_features = ['auto', 'sqrt']
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(10, 110, num = 11)]
max_depth.append(None)
# Minimum number of samples required to split a node
min_samples_split = [2, 5, 10]
# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2, 4]
# Method of selecting samples for training each tree
bootstrap = [True, False]
# Create the random grid
random_grid = {'n_estimators': n_estimators,
                  'max_features': max_features,
                 'max_depth': max_depth,
                 'min_samples_split': min_samples_split,
'min_samples_leaf': min_samples_leaf,
'bootstrap': bootstrap}
ran_model_1 = RandomizedSearchCV(estimator = rf_model, param_distributions = random_grid,
                                   n_iter = 100, cv = 3, verbose=2, random_state=42, n_jobs = -1)
ran_model_1.fit(X_train,y_train)
ran_model_1.best_params_
predictions=ran_model_1.best_estimator_.predict(X_test)
# Evaluating the Algorithm
#print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, predictions))
#print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, predictions)))
#score= ran_model_1.best_estimator_.score(X_test,y_test) #R2 Score
```

### 4. Saving the model

```
pickle.dump(ran_model_1,open('model.pkl','wb'))
model=pickle.load(open('model.pkl','rb'))
```

# 5. Deploying the model using Flask

#### App.py

#### ❖ Index.html

```
compared to the control of the contr
```

## Styling.css

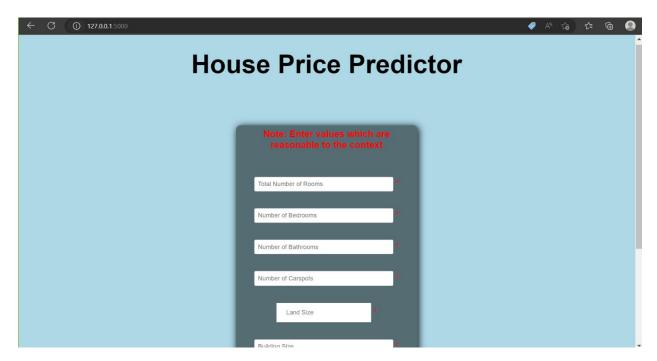
```
background-color: lightblue;
background-size: cover;
 form {
h1 {
           text-align: center;
font-size: 350%;
 button {
          font-weight: bold;
background-color: #4CAF50;
padding: 8px 16px;
display: inline-block;
          text-decoration: none;
border-radius:3px;
color: black;
border-color: black;
font-family: Arial;
          border-style: ridge;
  .para {
 .result {
          font-weight: bold;
background-color: #4CAF50;
          padding: 8px 16px;
display: inline-block;
          color: black;
border-color: black;
          font-family: Arial;
border-style: ridge;
 .Note {
           text-align: center;
         color: red;
margin: 7px;
padding: 7px 0px;
  }
.pred {
    text-align:center;
  .intro {
font-size: 20px;
/*Layout and structure of form body in page*/
div.container {
   background-color: rgba(0,0,0,0.5);
   font-size: 18px;
   margin: 1%;
   border-radius: 10px;
   border: 1px soid rgba)255,255,255,0.3);
   box-shadow: 2px 2px 15px
}
div.page {
width: 400px;
margin: 100px auto 0px auto;
 form.info {
    margin: 15px;
/*This section involves the design of the inputs in the form*/
input#Rooms {
  width: 300px;
  border: 1px solid #ddd;
  border-radius: 3px;
  outline: 0;
  padding: 7px;
  background-color: #fff;
  box-shadow: insert 1px 1px 5px rgba(0,0,0,0.3);
}
  }
input#Bedrooms {
    width: 300px;
    border: 1px solid #ddd;
```

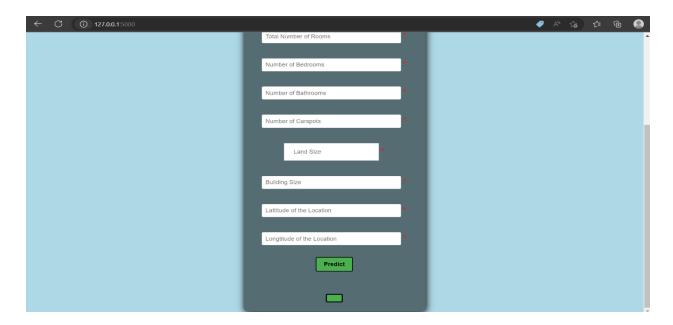
```
background-color: #fff;
box-shadow: insert 1px 1px 5px rgba(0,0,0,0.3);
input#Bathrooms {
         width: 300px;
         border: 1px solid #ddd;
         padding: 7px;
background-color: #fff;
box-shadow: insert 1px 1px 5px rgba(0,0,0,0.3);
input#Car {
   width: 300px;
          border: 1px solid #ddd;
          border-radius: 3px;
         padding: 7px;
background-color: #fff;
box-shadow: insert 1px 1px 5px rgba(0,0,0,0.3);
input#Land_Size {
         width: 300px;
         border: 1px solid #ddd;
         border-radius: 3px;
          padding: 7px;
         background-color: #fff;
box-shadow: insert 1px 1px 5px rgba(0,0,0,0.3);
input#Building_Area {
          width: 300px;
          border: 1px solid #ddd;
         background-color: #fff;
box-shadow: insert 1px 1px 5px rgba(0,0,0,0.3);
 input#Lattitude {
      width: 300px;
border: 1px solid #ddd;
border-radius: 3px;
outline: 0;
padding: 7px;
background-color: #fff;
box-shadow: insert 1px 1px 5px rgba(0,0,0,0.3);
 input#Longtitude {
      but#longittude {
    width: 300px;
    border: 1px solid #ddd;
    border-radius: 3px;
    outline: 0;
    padding: 7px;
    background-color: #fff;
    box-shadow: insert 1px 1px 5px rgba(0,0,0,0,3);
}
/*Design of links/buttons*/
a.link {
  font-weight: bold;
  background-color: orange;
  padding: 8px 16px;
  display: inline-block;
  text-decoration: none;
  border-radius:3px;
  color: black;
  border-color: black;
  border-style: ridge;
}
'*Responsible for shadow backgrounds*/
#menu-outer {
   height: 70px;
   background-color: rgba(0,0,0,0.5);
   font-size: 18px;
   margin: 1½;
   border-radius: 10px;
   border: 1px soid rgba)255,255,255,0.3);
}
 .table {
    display: table;
ul#horizontal-list {
       min-width: 696px;
list-style: none;
```

## Run the Flask Application

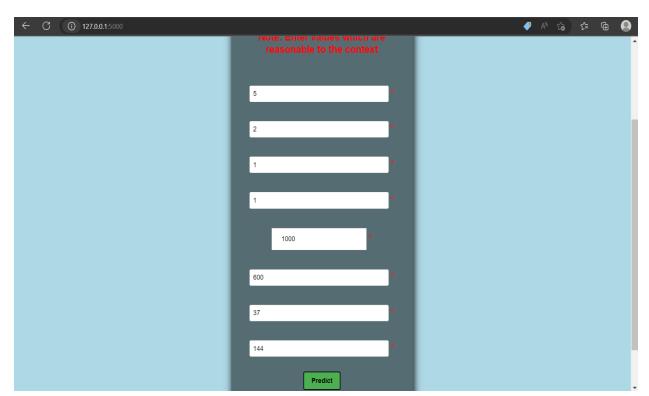


# Open the link in the browser





# Testing the App



# Getting the results

