Week 4

Name: Week 4 deployment
Report date: 9/27/24
Internship Batch: LISUM37
Version:1.0
Data intake by:Furkan Ay
Data intake reviewer:Data Glacier

Tabular data details: AmesHousing.csv

Total number of observations	2930
Total number of files	1
Total number of features	7
Base format of the file	csv
Size of the data	967 kb

Import Libraries

```
import pandas as pd
import pickle
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error
from sklearn.preprocessing import LabelEncoder
from flask import Flask, request, jsonify
```

Import the dataset

```
housing_data = pd.read_csv('AmesHousing.csv')
```

Clean the dataset

```
housing_data_cleaned = housing_data.drop(columns=['PID', 'Order'])
label_encoders = {}
for column in housing_data_cleaned.columns:
    if housing_data_cleaned[column].dtype == 'object':
        housing_data_cleaned[column].fillna(housing_data_cleaned[column].mode()[0], inplace=True)
        le = LabelEncoder()
        housing_data_cleaned[column] = le.fit_transform(housing_data_cleaned[column])
        label_encoders[column] = le
    else:
        housing_data_cleaned[column].fillna(housing_data_cleaned[column].median(), inplace=True)
```

Separate features and target

```
X = housing_data_cleaned.drop(columns=['SalePrice'])
y = housing_data_cleaned['SalePrice']
```

Building Model

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

filename = 'model.pkl'
with open(filename, 'wb') as file:
    pickle.dump(model, file)
```

```
y_pred = model.predict(X_test)
mae = mean_absolute_error(y_test, y_pred)
print(f"Mean Absolute Error: {mae}")
```

Deployment on Flask/ app.py

```python
from flask import Flask, request, jsonify
import numpy as np
import pickle


with open('model.pkl', 'rb') as file:
    model = pickle.load(file)


app = Flask(__name__)

@app.route('/predict', methods=['POST'])
def predict():
    data = request.get_json(force=True)
    features = np.array([list(data.values())])
    prediction = model.predict(features)
    return jsonify({'prediction': prediction[0]})

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