Operationalizing AI
Assignment 1: (Individual) Programming Assignment: House Price Prediction
Hiba Hassan (hibah)

AI Tools Interactions: Interactions

The primary and only AI tool utilized is ChatGPT to support code tuning, or understand the MLflow integrations within the code.

The following code chunks have segments that were tuned by chatgpt:

XGBoost Model

```
# Model 1: Zillow Kaggle Competition - XGboost
import xgboost as xgb
from sklearn.metrics import mean_absolute_error
import mlflow

# Define xgboost parameters
xgb_params = {
   'eta': 0.033,
   'max_depth': 6,
   'subsample': 0.80,
   'objective': 'reg:linear',
   'eval_metric': 'mae',
}

# Create DMatrix objects for train and test data
dtrain = xgb.DMatrix(X_train, label=y_train)
dtest = xgb.DMatrix(X_test)

# Cross-validation
cv_result = xgb.cv(xgb_params,
                   dtrain,
                   nfold=5,
                   num_boost_round=500,
                   early_stopping_rounds=5,
                   verbose_eval=10,
                   show_stdv=False
                  )
num_boost_rounds = len(cv_result)
print(num_boost_rounds)

# Train the XGBoost model
```

```
xgb_model = xgb.train(xgb_params, dtrain, num_boost_round=num_boost_rounds)

# Make predictions
y_pred = xgb_model.predict(dtest)

# Calculate evaluation metric (e.g., mean absolute error)
mae = mean_absolute_error(y_test, y_pred)
print("Mean Absolute Error:", mae)

# Log the model and its parameters with MLflow
with mlflow.start_run(tags={"Model": "XGBoost"}):

    # Log Metrics
    for key, value in xgb_params.items():
        mlflow.log_param(key, value)
    mlflow.log_metric("MAE", mae)

    # Log Model
    mlflow.xgboost.log_model(xgb_model, "xgboost_model")
```

Visualization

```
import matplotlib.pyplot as plt

# Create subplots
fig, axs = plt.subplots(1, 3, figsize=(15, 5))

# Bar plot for MAE
axs[0].bar(comparison_df['Model'], comparison_df['MAE'], color=['blue', 'orange', 'green'])
axs[0].set_title('Mean Absolute Error (MAE)')
axs[0].set_ylabel('MAE')

# Bar plot for MSE
axs[1].bar(comparison_df['Model'], comparison_df['MSE'], color=['blue', 'orange', 'green'])
axs[1].set_title('Mean Squared Error (MSE)')
axs[1].set_ylabel('MSE')

# Bar plot for R^2
axs[2].bar(comparison_df['Model'], comparison_df['R^2'], color=['blue', 'orange', 'green'])
axs[2].set_title('R^2 Score')
axs[2].set_ylabel('R^2')

# Adjust layout
plt.tight_layout()

# Show plots
```

plt.show()

Part 5: Model Serving

- Unsure about this section as chat GPT shares the following methodology but I am not sure how to connect and load the API/Json thing

Decision Trees #import requests #import json
Define the URL of the MLflow model server for the Decision Tree model #mlflow_server_url = "hhttp://127.0.0.1:5000/#/experiments/205159110639901886/runs/070dc76e36674c3ab63ec18e6a4d6ea0"
Make a POST request to the MLflow model server for prediction #response = requests.post(mlflow_server_url, json=sample_data)
Parse the prediction result #prediction_result = json.loads(response.text)
Print the prediction result #print("Prediction:", prediction_result)