MLOps - California Housing Price Prediction Report

1. Introduction

This report documents an end-to-end Machine Learning Operations (MLOps) project for predicting housing prices in California using the California Housing dataset. The project demonstrates key MLOps concepts, including data preprocessing, model training, hyperparameter tuning, evaluation, and deployment. The goal was to build a robust pipeline that preprocesses data, trains a K-Nearest Neighbors (KNN) regressor, optimizes hyperparameters, and deploys the model via a FastAPI web service.

Key Objectives:

- Load and explore the California Housing dataset.
- Preprocess data (imputation, scaling) using `ColumnTransformer`.
- Train a KNN regressor with hyperparameter tuning using `GridSearchCV`.
- Evaluate model performance using R^2 score, MSE, and RMSE.
- Save the trained model and deploy it as a FastAPI endpoint.

2. Task Completion

2.1 Data Loading and Exploration

The dataset was loaded using `fetch_california_housing` from scikit-learn. It contains 20,640 samples with 8 features and a target variable (median house value in $100,000s).

Dataset Summary:

Feature	Description	Example Values
`MedInc`	Median income in block group	8.3252
`HouseAge`	Median house age in block group	41.0
`AveRooms`	Avg. number of rooms per household	6.984127
`AveBedrms`	Avg. number of bedrooms per household	1.023810
`Population`	Block group population	322.0
`AveOccup`	Avg. number of household members	2.555556
`Latitude`	Block group latitude	37.88
`Longitude`	Block group longitude	-122.23

Target Variable:

- **`MedHouseVal`:** Median house value (in $100,000s)
- **Mean:** 2.068
- **Range:** 0.14999 to 5.00001

2.2 Data Preprocessing & Pipeline Construction

A preprocessing pipeline was built to:

1. Handle missing values using `SimpleImputer` (mean imputation).

2. Standardize features using `StandardScaler`.
3. Combine transformations using `ColumnTransformer`.

```
numeric_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='mean')),
    ('scaler', StandardScaler())
])

preprocessor = ColumnTransformer(transformers=[
    ('num', numeric_transformer, numeric_features)
])

pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('knn', KNeighborsRegressor())
])
```

2.3 Hyperparameter Tuning with GridSearchCV

A 5-fold cross-validation was performed to optimize:
- `n_neighbors`: [3, 5, 7, 9]
- `weights`: ['uniform', 'distance']
- `p` (distance metric): [1 (Manhattan), 2 (Euclidean)]

Best Parameters Found:

```
{'knn__n_neighbors': 9, 'knn__p': 1, 'knn__weights': 'distance'}
```

Best CV R^2 Score: 0.731

2.4 Model Evaluation

The best model was evaluated on the test set (20% split):

Metric	**Score**
R^2	0.722
MSE	0.364
RMSE	0.603

Interpretation:
- The model explains 72.2% of the variance in housing prices.
- The RMSE of 0.603 (in $100,000s) means predictions are typically off by ~$60,300.

2.5 Model Saving & Deployment

The trained pipeline was saved as a `.pkl` file:

```
with open('california_knn_pipeline.pkl', 'wb') as f:
    pickle.dump(best_model, f)
```

FastAPI Deployment (Optional)
A `/predict` endpoint was created to serve predictions:

```
from fastapi import FastAPI
import pickle
import pandas as pd

app = FastAPI()

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

@app.post("/predict")
def predict(data: dict):
    input_data = pd.DataFrame([data])
    prediction = model.predict(input_data)[0]
    return {"predicted_price": prediction  100000}   Convert to dollars
```

Example API Request:

```
curl -X POST "http://127.0.0.1:8000/predict" \
-H "Content-Type: application/json" \
-d '{"MedInc":8.3, "HouseAge":41, "AveRooms":6.9, "AveBedrms":1.02, "Population":322,
"AveOccup":2.55, "Latitude":37.88, "Longitude":-122.23}'
```

Expected Output:

```
{"predicted_price": 420000.0}
```

3. Conclusion & Learnings

Key Takeaways:

1. MLOps Workflow:

- Successfully implemented a complete ML pipeline from data loading to deployment.
- Demonstrated the importance of preprocessing, hyperparameter tuning, and evaluation.

2. Model Performance:

- The KNN regressor achieved ~72% R^2, indicating decent predictive power.
- Hyperparameter tuning significantly improved performance (from default R^2 ~0.65 to 0.722).

3. Future Improvements:

- Try more advanced models (Random Forest, Gradient Boosting).
- Feature engineering (e.g., interaction terms, geographic clustering).
- Deploy in the cloud (AWS, GCP) for scalability.

Final Thoughts

This project provided hands-on experience in MLOps best practices, including reproducible pipelines, model evaluation, and API deployment. The model could be further improved, but the current implementation serves as a strong baseline for housing price prediction.

GitHub/Colab Link: [Link]
Submitted By: Daniel Muthama
Date: 11/7/2025