California Housing Price Predictor Documentation

By 22i-2301,Ahmed Mustafa

Introduction

The **California Housing Price Predictor** is a machine learning web application built using **Flask**, which predicts the median house value in California based on various features. This project aims to apply machine learning techniques to real-world data, allowing users to predict housing prices by selecting a model and providing feature inputs.

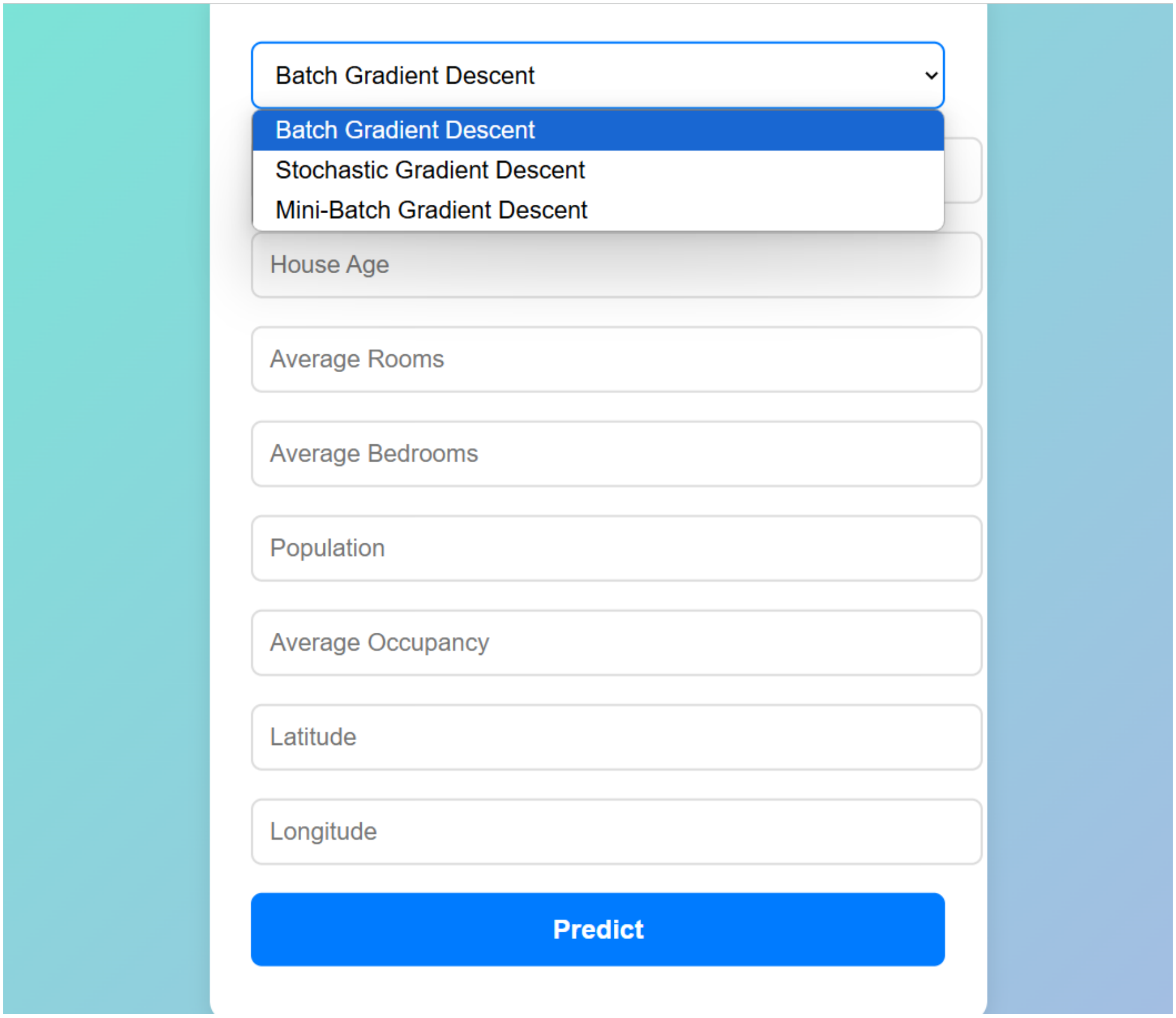
Objective

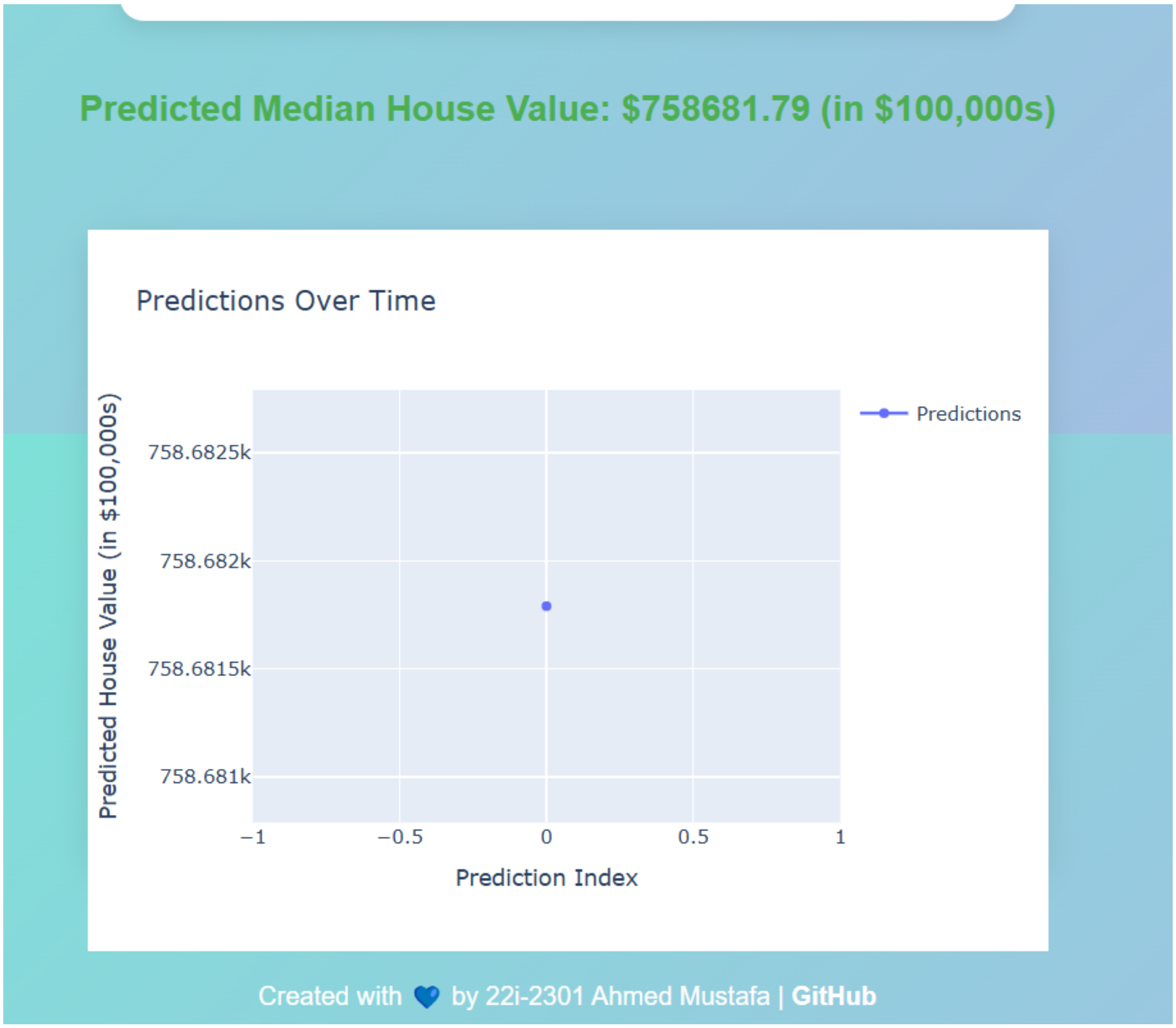
The goal of this project is to use machine learning to predict the median house value of California based on input features such as:

- Median income
- House age
- Average number of rooms
- Average number of bedrooms
- Population
- Latitude and Longitude

Three distinct **Gradient Descent** variants were implemented to solve the regression problem, namely:

- **Batch Gradient Descent (BGD)**
- **Stochastic Gradient Descent (SGD)**
- **Mini-Batch Gradient Descent (MBGD)**


Batch Gradient Descent
Batch Gradient Descent
Stochastic Gradient Descent
Mini-Batch Gradient Descent
House Age
Average Rooms
Average Bedrooms
Population
Average Occupancy
Latitude
Longitude
Predict



This approach demonstrates how different optimization algorithms can be applied to improve prediction accuracy.

Machine Learning Workflow

1. Data Collection and Preprocessing

The dataset used for this project is a modified version of the **California housing dataset**. It contains information about various features of homes in California, such as:

- **Median Income**: Average income of households in the area.
- **House Age**: The median age of the houses in the region.
- **Average Rooms**: The average number of rooms in homes.

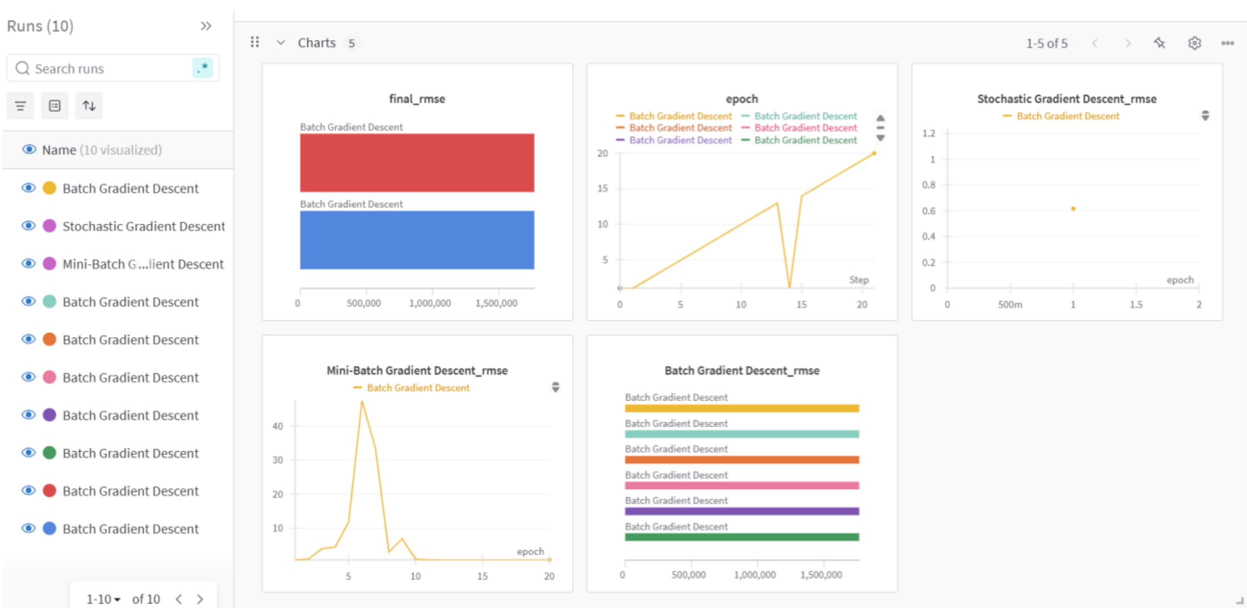
- **Average Bedrooms**: The average number of bedrooms in homes.
- **Population**: Population of the area.
- **Latitude and Longitude**: Geographical location.

Preprocessing Steps:

- **Normalization/Scaling**: The data is normalized to ensure that all features are on a similar scale, making the optimization process more efficient.
- **Train-Test Split**: The dataset is divided into training and test sets (80% training, 20% test).
- **Missing Value Handling**: Any missing or incomplete data is handled appropriately, either by filling or removing such entries.

2. Model Selection

For this task, three different types of **Gradient Descent** optimization algorithms were used to train the models:

- **Batch Gradient Descent (BGD)**: Updates parameters after processing the entire training dataset.
- **Stochastic Gradient Descent (SGD)**: Updates parameters after processing each individual training example.
- **Mini-Batch Gradient Descent (MBGD)**: A compromise between BGD and SGD, where updates are made after processing a subset of the training data.
- 

Each of these algorithms was used to train a linear regression model, as the problem involves predicting a continuous target value (the median house price).

3. Training the Models

Each model is trained using the selected algorithm. Here's a breakdown of the training process:

1. **Model Initialization**: Start with random values for the parameters (weights and bias).
2. **Cost Function**: Use **Mean Squared Error (MSE)** as the cost function to evaluate how well the model is performing.
3. **Optimization**: Use the respective Gradient Descent algorithm to minimize the cost function and update the model parameters.
4. **Evaluation**: After training, evaluate the model's performance using the test set. The primary metric used for evaluation is **Mean Absolute Error (MAE)**, which measures the average absolute difference between predicted and actual values.

Model Evaluation

1. Performance Metrics

To evaluate the models, several metrics are considered:

- **Mean Squared Error (MSE)**: Helps to understand the variance in errors.
- **R-squared (R^2)**: Indicates the proportion of variance in the dependent variable that is predictable from the independent variables.

2. Model Comparison

The models trained using different variants of Gradient Descent are compared to determine which one provides the best results:

- **BGD**: Suitable for large datasets but slower in convergence.
- **SGD**: Faster for large datasets but may have higher variance in updates.
- **MBGD**: Strikes a balance between BGD and SGD, providing faster convergence without sacrificing accuracy.

Deployment

1. GitHub Repository

The full code and documentation for this project are available in the GitHub repository. The repository contains:

- **Model Training Scripts**: Code to train and evaluate the models.
- **Preprocessing Scripts**: Code for data preprocessing and scaling.
- **Web Application**: Flask application to serve the model and predict values based on user inputs.

You can access the repository here:
https://github.com/Mustafaahmed10/MachineLearning_Assignment3

2. Hugging Face Model Link

For easy access to the trained models, the models have been uploaded to **Hugging Face**. The models are available for download, ensuring that the trained models can be easily integrated into the Flask application for inference.

You can access the models here:

- BGD Model
- SGD Model
- MBGD Model
- LINK:
- https://huggingface.co/i222301ahmedmustafa/california-housing-regressor/tree/main

3. Inference Script

The inference script allows for predictions to be made using the trained models. Here's how to use the script:

1. **Wandb.ai link**: https://wandb.ai/i222301-national-university-of-computer-and-emerging-sci/gradient-descent-comparison?nw=nwuseri222301
2. **Run Inference**:
   - Load the model from Hugging Face or from a local file.
   - Pass the input features (e.g., median income, house age) to the model.
   - The model will return the predicted house price.

Example:

```
from joblib import load

# Load the model
model = load("bgd_model.pkl")

# Example input features
features = [5.0, 20, 6, 3, 1000, 4, 37.77, -122.42]  # Replace with actual input values

# Predict the median house value
prediction = model.predict([features])
print(f"Predicted House Value: {prediction}")
```

4. Web App

The web application provides a user-friendly interface for interacting with the model. Using **Flask** for my app development.

Conclusion

This project demonstrates the application of machine learning techniques, particularly **Gradient Descent**, to predict housing prices in California. By providing three distinct optimization algorithms, users can observe how different approaches impact the model's performance.

The **Flask web application** serves as the interface for making predictions, **Wandb.ai** serves as the predictions inference , while the **Hugging Face model repository** ensures that the trained models are easily accessible for future use.
