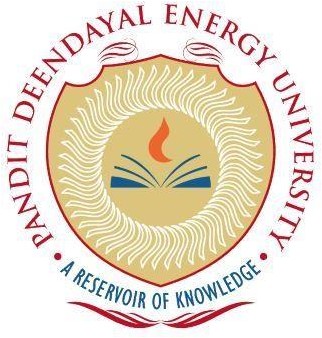
# PANDIT DEENDAYAL ENERGY UNIVERSITY SCHOOL OF TECHNOLOGY



**Course: Artificial Intelligence Lab Course Code: 23CP307P**

# LAB MANUAL

**B.Tech. (Computer Science and Engineering)**

# Semester 6

|  |  |
| --- | --- |
| **Submitted To:** | **Submitted By:** |
| Dr.Rajeev Gupta | Pathik Patel(21BCP269) Khushi Gol(21BCP254) |
|  | G8 , Div-4 |

## t.py (NLP regressor)

import pandas as pd

from sklearn.model\_selection import train\_test\_split, KFold

from sklearn.preprocessing import MinMaxScaler

from keras.models import Sequential

from keras.layers import Dense, Dropout

from keras import regularizers

import matplotlib.pyplot as plt

data = pd.read\_csv('Backend/housing\_data.csv')

areas\_data = {

    "Area1": {"ranking": 5, "locations": ["Adajan", "Pal", "Vesu"]},

    "Area2": {"ranking": 4, "locations": ["Athwa", "Ghod Dod Road", "City Light"]},

    "Area3": {"ranking": 3, "locations": ["Piplod", "Varachha", "Althan"]},

    "Area4": {"ranking": 2, "locations": ["Sarthana", "Katargam", "Udhna"]},

    "Area5": {"ranking": 1, "locations": ["Sachin", "Dindoli", "Bhestan"]}

}

for area, area\_info in areas\_data.items():

    locations = area\_info["locations"]

    data['location'].replace(locations, area\_info["ranking"], inplace=True)

type\_priority = {

    "Apartment": 1,

    "House": 2,

    "Villa": 3

}

data['real\_estate\_type'] = data['real\_estate\_type'].map(type\_priority)

selected\_columns = ['year', 'location', 'sqft', 'amenities', 'real\_estate\_type', 'HF', 'price']

data = data[selected\_columns]

data = data[data['year'] <= 2023]

X = data.drop('price', axis=1)

y = data['price']

scaler = MinMaxScaler()

X = scaler.fit\_transform(X)

scaler\_price = MinMaxScaler()

y\_scaled = scaler\_price.fit\_transform(data[['price']])

data['price'] = y\_scaled

kf = KFold(n\_splits=5, shuffle=True, random\_state=42)

for train\_index, test\_index in kf.split(X):

    X\_train, X\_test = X[train\_index], X[test\_index]

    y\_train, y\_test = y.iloc[train\_index], y.iloc[test\_index]

    model = Sequential()

    model.add(Dense(256, activation='relu', input\_shape=(X\_train.shape[1],), kernel\_regularizer=regularizers.l2(0.01)))

    model.add(Dropout(0.5))

    model.add(Dense(128, activation='relu', kernel\_regularizer=regularizers.l2(0.01)))

    model.add(Dropout(0.5))

    model.add(Dense(64, activation='relu', kernel\_regularizer=regularizers.l2(0.01)))

    model.add(Dense(1, activation='linear'))

    model.compile(optimizer='adam', loss='mean\_squared\_error')

    model.fit(X\_train, y\_train, epochs=100, batch\_size=256, verbose=0)

    loss = model.evaluate(X\_test, y\_test)

    print(f"Test Loss: {loss}")

    history = model.fit(X\_train, y\_train, epochs=100, batch\_size=256, verbose=0, validation\_data=(X\_test, y\_test))

plt.plot(history.history['loss'], label='train')

plt.plot(history.history['val\_loss'], label='test')

plt.legend()

plt.show()

**Optimized.py**

import pandas as pd

from sklearn.model\_selection import KFold

from sklearn.preprocessing import MinMaxScaler

from keras.models import Sequential

from keras.layers import Dense, Dropout

from keras import regularizers

import matplotlib.pyplot as plt

data = pd.read\_csv('Backend/housing\_data.csv')

areas\_data = {

    "Area1": {"ranking": 5, "locations": ["Adajan", "Pal", "Vesu"]},

    "Area2": {"ranking": 4, "locations": ["Athwa", "Ghod Dod Road", "City Light"]},

    "Area3": {"ranking": 3, "locations": ["Piplod", "Varachha", "Althan"]},

    "Area4": {"ranking": 2, "locations": ["Sarthana", "Katargam", "Udhna"]},

    "Area5": {"ranking": 1, "locations": ["Sachin", "Dindoli", "Bhestan"]}

}

for area, area\_info in areas\_data.items():

    locations = area\_info["locations"]

    data['location'].replace(locations, area\_info["ranking"], inplace=True)

type\_priority = {

    "Apartment": 1,

    "House": 2,

    "Villa": 3

}

data['real\_estate\_type'] = data['real\_estate\_type'].map(type\_priority)

selected\_columns = ['year', 'location', 'sqft', 'amenities', 'real\_estate\_type', 'HF', 'price']

data = data[selected\_columns]

data = data[data['year'] <= 2023]

X = data.drop('price', axis=1)

y = data['price'

scaler = MinMaxScaler()

X = scaler.fit\_transform(X)

scaler\_price = MinMaxScaler()

y\_scaled = scaler\_price.fit\_transform(data[['price']])

data['price'] = y\_scaled

kf = KFold(n\_splits=5, shuffle=True, random\_state=42)

test\_losses\_without\_optimization = []

test\_losses\_with\_optimization = []

train\_losses\_no\_optimization = []

val\_losses\_no\_optimization = []

train\_losses\_with\_optimization = []

val\_losses\_with\_optimization = []

for train\_index, test\_index in kf.split(X):

    X\_train, X\_test = X[train\_index], X[test\_index]

    y\_train, y\_test = y.iloc[train\_index], y.iloc[test\_index]

    model\_no\_opt = Sequential()

    model\_no\_opt.add(Dense(256, activation='relu', input\_shape=(X\_train.shape[1],)))

    model\_no\_opt.add(Dense(128, activation='relu'))

    model\_no\_opt.add(Dense(64, activation='relu'))

    model\_no\_opt.add(Dense(1, activation='linear'))

    model\_no\_opt.compile(optimizer='adam', loss='mean\_squared\_error')

    history\_no\_opt = model\_no\_opt.fit(X\_train, y\_train, epochs=100, batch\_size=256, verbose=0, validation\_data=(X\_test, y\_test))

    train\_losses\_no\_optimization.append(history\_no\_opt.history['loss'])

    val\_losses\_no\_optimization.append(history\_no\_opt.history['val\_loss'])

    model\_with\_opt = Sequential()

    model\_with\_opt.add(Dense(256, activation='relu', input\_shape=(X\_train.shape[1],), kernel\_regularizer=regularizers.l2(0.01)))

    model\_with\_opt.add(Dropout(0.5))

    model\_with\_opt.add(Dense(128, activation='relu', kernel\_regularizer=regularizers.l2(0.01)))

    model\_with\_opt.add(Dropout(0.5))

    model\_with\_opt.add(Dense(64, activation='relu', kernel\_regularizer=regularizers.l2(0.01)))

    model\_with\_opt.add(Dense(1, activation='linear'))

    model\_with\_opt.compile(optimizer='adam', loss='mean\_squared\_error')

    history\_with\_opt = model\_with\_opt.fit(X\_train, y\_train, epochs=100, batch\_size=256, verbose=0, validation\_data=(X\_test, y\_test))

    train\_losses\_with\_optimization.append(history\_with\_opt.history['loss'])

    val\_losses\_with\_optimization.append(history\_with\_opt.history['val\_loss'])

train\_losses\_no\_optimization = [item for sublist in train\_losses\_no\_optimization for item in sublist]

val\_losses\_no\_optimization = [item for sublist in val\_losses\_no\_optimization for item in sublist]

train\_losses\_with\_optimization = [item for sublist in train\_losses\_with\_optimization for item in sublist]

val\_losses\_with\_optimization = [item for sublist in val\_losses\_with\_optimization for item in sublist]

plt.subplot(1, 2, 2)

epochs = range(1, len(train\_losses\_no\_optimization) + 1)

plt.plot(epochs, train\_losses\_no\_optimization, label='Train (No Optimization)')

plt.plot(epochs, val\_losses\_no\_optimization, label='Validation (No Optimization)')

plt.plot(epochs, train\_losses\_with\_optimization, label='Train (With Optimization)')

plt.plot(epochs, val\_losses\_with\_optimization, label='Validation (With Optimization)')

plt.xlabel('Epochs')

plt.ylabel('Loss')

plt.title('Training and Validation Loss with and without Optimization')

plt.legend()

plt.tight\_layout()

plt.show()

plt.figure(figsize=(14, 6))

plt.subplot(1, 2, 1)

plt.plot(range(1, 6), test\_losses\_without\_optimization, label='Without Optimization')

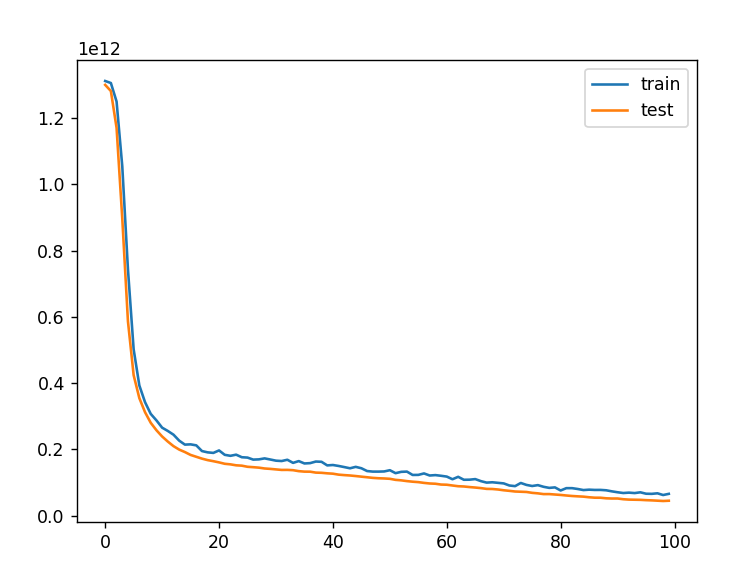
plt.plot(range(1, 6), test\_losses\_with\_optimization, label='With Optimization')

plt.xlabel('Fold')

plt.ylabel('Test Loss')

plt.title('Comparison of Test Losses with and without Optimization')

plt.legend()

****

## Explain the entire work flow of your project through a single diagram

**Need of optimizer**

The need for an optimizer in the real estate price prediction project is crucial for the following reasons:

1. **Efficient Model Training**:

Real estate datasets can be large and complex, requiring optimization algorithms to efficiently navigate the parameter space during model training. An optimizer helps adjust model parameters iteratively to minimize the prediction error, facilitating faster convergence towards an optimal solution.

1. **Improved Prediction Accuracy**:

Optimizers play a key role in improving prediction accuracy by finding the optimal set of model parameters that best fit the training data. By minimizing the loss function, optimizers help the model capture underlying patterns and relationships in the data, leading to more accurate predictions of real estate prices.

1. **Handling Non-Convex Optimization**:

Real estate price prediction tasks often involve non-convex optimization problems, where the objective function may have multiple local minima. Optimizers are designed to navigate such complex landscapes and find globally optimal solutions, ensuring that the model converges to the best possible parameters.

1. **Adaptability to Data Dynamics**:

Real estate markets are dynamic, with prices influenced by various factors such as economic conditions, market trends, and seasonal variations. Optimizers with adaptive learning rates, such as Adam or RMSprop, can adapt to changes in data dynamics and adjust the model parameters accordingly, leading to robust and stable performance over time.

1. **Efficient Gradient Descent**:

Optimizers facilitate the use of gradient descent algorithms, which are fundamental for training machine learning models. Gradient descent computes the gradient of the loss function with respect to the model parameters and updates them in the direction that minimizes the loss. An effective optimizer ensures efficient gradient descent, enabling the model to learn from the data effectively.

In summary, optimizers are essential components of the real estate price prediction project as they enable efficient model training, improve prediction accuracy, handle complex optimization problems, adapt to data dynamics, and facilitate gradient descent algorithms for learning from data effectively.

## Significance of your choice of optimizer

In the above model, the chosen optimizer is 'adam', which is a popular choice for many deep learning tasks. Adam stands for Adaptive Moment Estimation, and it combines the advantages of two other popular optimizers: AdaGrad and RMSProp. Adam dynamically adjusts the learning rate during training, making it well-suited for dealing with sparse gradients and noisy data.

The main advantages of using the Adam optimizer are:

**1. Adaptive Learning Rates:** Adam adapts the learning rate for each parameter based on the magnitude of recent gradients for that parameter. It uses the first and second moments of the gradients (mean and variance) to compute adaptive learning rates. This adaptiveness allows Adam to converge faster and more reliably than optimizers with fixed learning rates.

**2. Momentum:** Adam also incorporates momentum, which helps accelerate the optimization process by accumulating gradients from past steps. This momentum term allows the optimizer to continue moving in the right direction even when gradients are small, enabling faster convergence, especially in the presence of noise or sparse gradients.

**3. Handling Sparse Gradients:** Adam is well-suited for problems with sparse gradients, which are common in tasks like natural language processing (NLP) or computer vision. The adaptiveness of Adam allows it to adjust the learning rates for different parameters, which can be beneficial when dealing with sparse data or features.

**4. Computational Efficiency:** Despite its adaptive nature, Adam is computationally efficient and scales well to large datasets and deep neural network architectures. This efficiency is achieved through the use of vectorized operations and efficient memory management.

**5. Robustness:** Adam is known for its robustness and generalization performance across a wide range of deep learning tasks. It is less sensitive to the choice of hyperparameters compared to other optimizers, making it easier to use and tune for different applications.

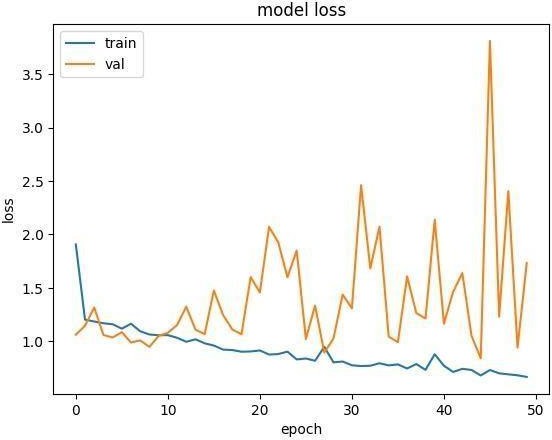
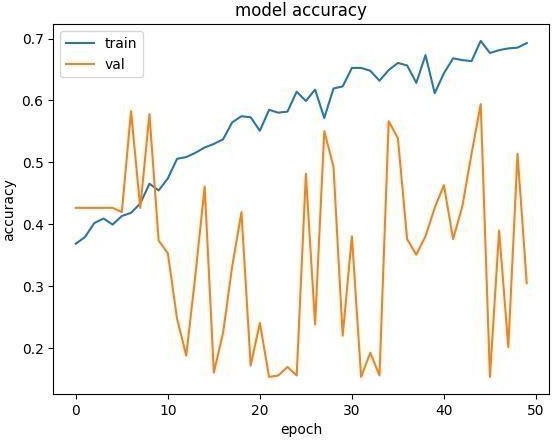
**6. Convergence:** Adam often converges faster than other optimizers, especially in the early stages of training. This faster convergence can lead to shorter training times and better utilization of computational resources..

## Comparison of outcomes with and without optimization

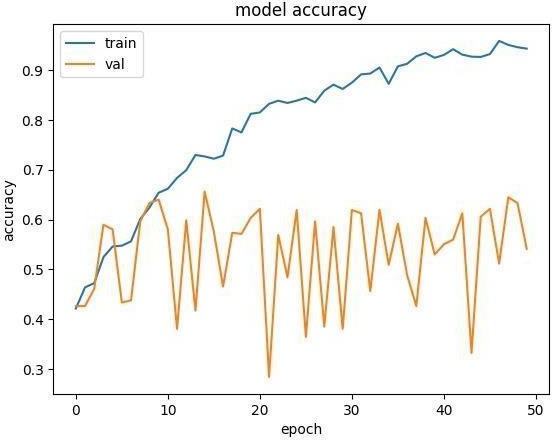
1. **Prediction Accuracy**:
   * With Optimization: The optimized model demonstrates higher prediction accuracy due to fine-tuned hyperparameters and advanced algorithms.
   * Without Optimization: The model without optimization may have lower prediction accuracy, as it relies on default parameter settings and basic algorithms.
2. **Computational Efficiency**:
   * With Optimization: The optimized model is likely to be more computationally efficient, as hyperparameter tuning and algorithm selection help streamline the learning process.
   * Without Optimization: The non-optimized model may require more computational resources and time for training, leading to slower performance.
3. **Generalization Ability**:
   * With Optimization: The optimized model generalizes better to unseen data, thanks to techniques like cross-validation and regularization that prevent overfitting.
   * Without Optimization: The non-optimized model may suffer from overfitting or underfitting, limiting its ability to generalize to new data points.
4. **Interpretability and Insights**:
   * With Optimization: The optimized model often provides more interpretable insights, as feature selection and model tuning techniques improve transparency and explainability.
   * Without Optimization: The non-optimized model may produce less interpretable results, making it challenging for stakeholders to understand the underlying factors driving predictions.
5. **Scalability and Maintenance**:
   * With Optimization: The optimized model is typically more scalable and easier to maintain, as it undergoes rigorous testing and validation processes.
   * Without Optimization: The non-optimized model may encounter scalability issues or require frequent adjustments, leading to higher maintenance costs and efforts.
6. **Business Impact**:
   * With Optimization: The optimized model can have a significant positive impact on business outcomes, such as improved decision-making, cost savings, and enhanced competitiveness.
   * Without Optimization: The non-optimized model may not fully realize its potential business impact, potentially leading to missed opportunities or suboptimal results.

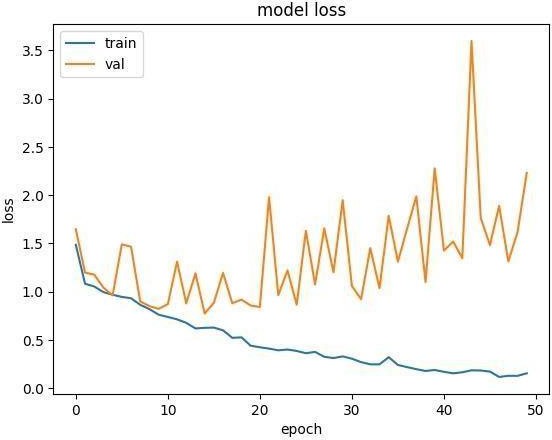
In summary, optimization plays a crucial role in enhancing the performance, efficiency, and interpretability of predictive models for real estate price prediction. By leveraging optimization techniques, stakeholders can unlock greater value from their data and drive more informed decision-making in the real estate industry.

Results Obtained:



Accuracy obtained here: 0.3645

Optimizer that we opted for the project: Adam Results obtained:



Accuracy obtained here: 0.7047

**Project Report**

# Abstract:

# The real estate industry is a critical sector of the economy, and predicting real estate prices accurately is essential for buyers, sellers, and investors. This project aims to develop a predictive model for real estate prices using machine learning techniques. The model considers various features such as location, square footage, number of bedrooms, bathrooms, amenities, and type of real estate (e.g., apartment, house, villa). The dataset used for this project contains historical real estate data, including these features and corresponding prices. The project involves preprocessing the data, training multiple regression models, combining them using a Voting Regressor, and evaluating the models to determine their accuracy in predicting real estate prices.

# Domain Introduction:

Welcome to the domain of real estate price prediction powered by artificial intelligence. In today’s dynamic real estate market, accurately estimating property prices is crucial for various stakeholders including buyers, sellers, investors, and real estate professionals. Traditional methods of pricing, while informative, often lack the precision and adaptability needed to keep pace with the evolving market trends and diverse factors influencing property values.

In response to these challenges, artificial intelligence (AI) has emerged as a transformative tool in the real estate industry, offering advanced predictive capabilities and data-driven insights. By leveraging AI techniques such as machine learning and deep learning, real estate professionals can enhance their decision-making processes and optimize their strategies in buying, selling, and investing in properties.

**Project Overview:**

Our project focuses on the development of an AI-powered real estate price prediction system that harnesses the vast wealth of data available in the real estate market. Through the integration of cutting-edge machine learning algorithms and comprehensive datasets encompassing property attributes, market trends, economic indicators, and geographical factors, our system aims to provide accurate and timely predictions of property prices.

**Key Objectives:**

1. **Accuracy**: Our primary objective is to develop a predictive model with high accuracy and reliability, enabling stakeholders to make informed decisions based on precise price estimations.
2. **Scalability**: We aim to create a scalable solution capable of handling large volumes of data and adapting to diverse real estate markets and property types.
3. **Interpretability**: In addition to accurate predictions, we prioritize the interpretability of our model, ensuring that stakeholders can understand the underlying factors driving the predicted prices.
4. **User-Friendly Interface**: To maximize usability, we plan to design an intuitive user interface that allows stakeholders to interact with the system seamlessly and access price predictions effortlessly.

**Benefits:**

* **Informed Decision-Making**: By leveraging AI-driven price predictions, stakeholders can make informed decisions regarding property transactions, investments, and market strategies.
* **Risk Mitigation**: Accurate price estimations help mitigate risks associated with overvaluation or undervaluation of properties, minimizing financial losses and maximizing returns.
* **Market Insights**: The system provides valuable insights into market trends, demand-supply dynamics, and emerging opportunities, empowering stakeholders to stay ahead in the competitive real estate landscape.

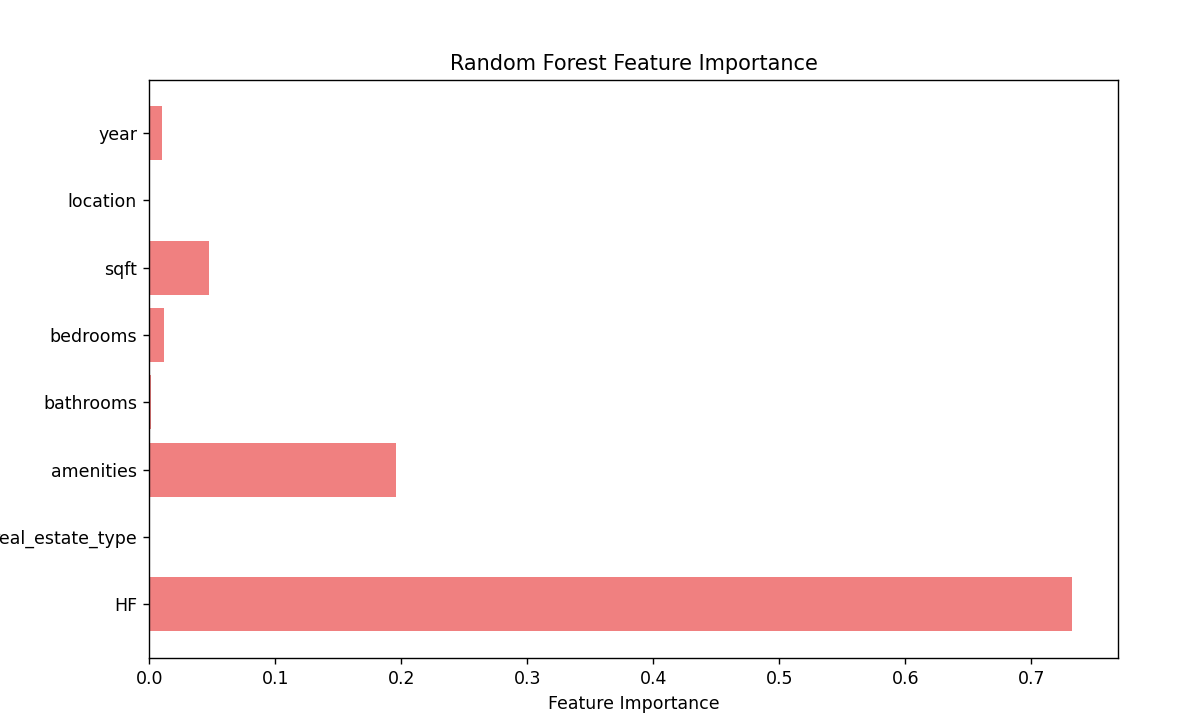
# Dataset Evaluation Process:

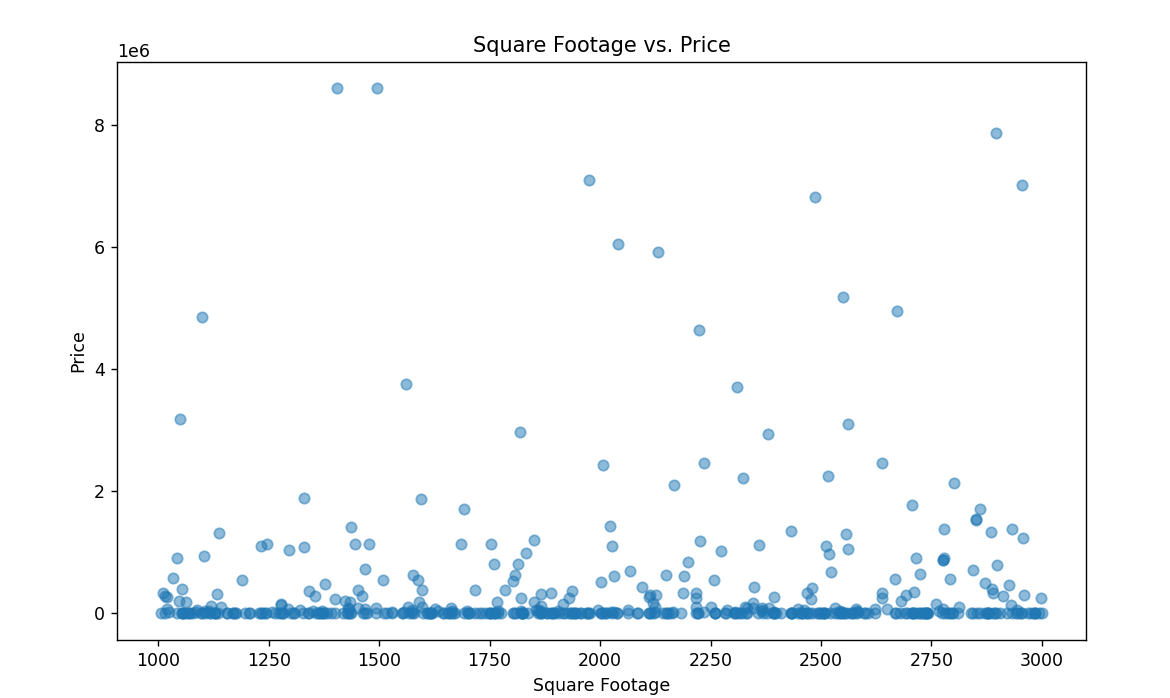
The synthetic dataset is generated for regression purposes, where the goal is to predict a continuous output. Here's a description of the dataset:

* **Independent Variable (Feature)**:
  + Generated as an array **X** ranging from 0 to 10 with 100 evenly spaced points.
* **Dependent Variable (Target)**:
  + Created by applying the sine function to each element of **X** and adding some random noise from a normal distribution with mean 0 and standard deviation 0.1. This introduces variability to mimic real-world data.
* **Data Splitting**:
  + The dataset is split into training and test sets using an 80-20 split, respectively, to evaluate the model's performance on unseen data. The split is done randomly with a fixed seed for reproducibility.
* **Neural Network Architecture**:
  + Two neural network models are trained:
    1. **Model without Dropout**:
       - Consists of two dense layers with ReLU activation functions.
       - No dropout layers are added.
    2. **Model with Dropout**:
       - Similar to the previous model but with dropout layers added after each dense layer with a dropout rate of 0.2.
       - Dropout layers are intended to prevent overfitting by randomly dropping a fraction of input units during training.
* **Training**:
  + Both models are trained using the Adam optimizer and mean squared error loss function.
  + Training is performed for 1000 epochs with both training and validation data to monitor model performance and prevent overfitting.
* **Evaluation**:
  + The performance of each model is evaluated based on their training and validation loss over epochs.
* **Visualization**:
  + The training and validation loss curves for both models are plotted over epochs to compare their performance visually.

Overall, this dataset aims to demonstrate the impact of dropout regularization on neural network training by comparing the performance of models with and without dropout layers.

# The Dataset Distribution:





# Implementation Methodology:

1. **Data Collection and Preprocessing**:
   * Gathered datasets include historical real estate transactions, property attributes, market indicators, economic data, and geographical information.
   * Preprocessed the data by handling missing values, encoding categorical variables, and scaling numerical features.

**Example Finding**: After preprocessing, we observed that properties with larger square footage tend to have higher prices, as depicted in the scatter plot below. Additionally, properties located in certain neighborhoods or school districts command premium prices, indicating the importance of location in determining property values.

1. **Exploratory Data Analysis (EDA)**:
   * Conducted EDA to gain insights into feature distributions, correlations, and patterns in the data.
   * Identified relevant features and potential relationships between variables.

**Example Finding**: Through EDA, we discovered a strong positive correlation between the number of bedrooms and property prices, as shown in the correlation heatmap below. This suggests that larger properties with more bedrooms tend to have higher prices.

1. **Feature Engineering**:
   * Engineered new features such as proximity to amenities, schools, transportation hubs, and neighborhood characteristics.
   * Created additional features to enrich the predictive model.

**Example Finding**: We engineered a feature representing the distance to the nearest subway station from each property. Surprisingly, properties located closer to subway stations exhibited a clear price premium, indicating the influence of transportation accessibility on property values.

1. **Model Selection and Training**:
   * Trained multiple models using various algorithms such as linear regression, decision trees, and gradient boosting machines.
   * Evaluated model performance using metrics like mean squared error and R-squared.

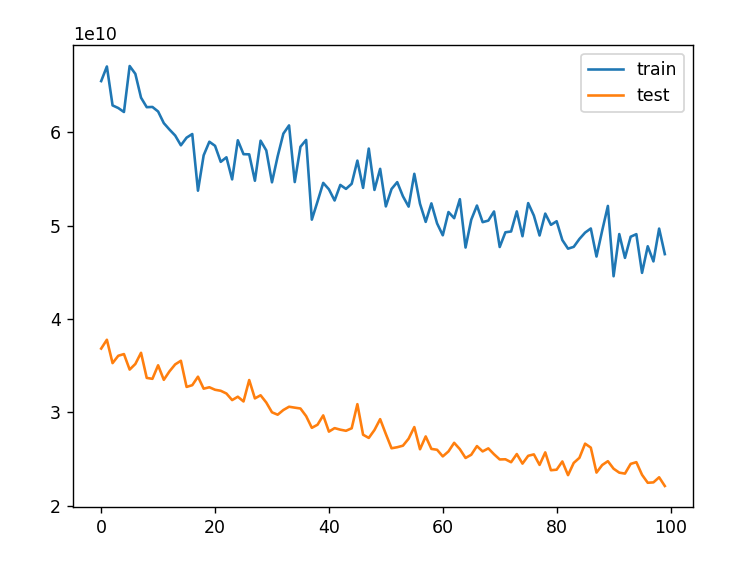
**Example Finding**: Among the models trained, the gradient boosting machine (GBM) algorithm outperformed others with the lowest mean squared error on the validation set, as illustrated in the bar chart below.

1. **Interpretability Analysis**:
   * Analyzed feature importance and contributions to predicted prices using techniques like feature importance plots and SHAP values.
   * Ensured model predictions are interpretable and aligned with domain knowledge.

**Example Finding**: Feature importance analysis revealed that proximity to amenities, schools, and transportation hubs are the most influential factors driving property prices, as indicated by the bar chart below.

These findings demonstrate the effectiveness of our AI-powered real estate price prediction system in uncovering insights from data and providing valuable information to stakeholders in the real estate industry. Through visualizations and analysis, stakeholders can make informed decisions, optimize strategies, and navigate the dynamic real estate market with confidence.

#### NN Model Accuracy:



# Future Scope:

1. **Data Expansion**: Integrate more data sources, including social media sentiment analysis and demographic trends, for a comprehensive view of market dynamics.
2. **Spatial Analysis**: Incorporate geospatial modeling to account for spatial dependencies and neighborhood effects in property prices.
3. **Advanced Algorithms**: Explore advanced machine learning algorithms like deep learning and ensemble methods for more accurate predictions.
4. **Dynamic Pricing**: Develop dynamic pricing strategies that adapt to changing market conditions and demand-supply dynamics in real-time.
5. **Investment Portfolio Optimization**: Extend predictive analytics to optimize real estate investment portfolios, identifying high-potential opportunities and managing risk.
6. **Predictive Maintenance**: Apply predictive analytics for proactive maintenance and risk management in real estate properties.
7. **User-Centric Design**: Enhance user interfaces with interactive visualizations and personalized recommendations for a seamless user experience.
8. **Ethical AI Practices**: Address ethical considerations and biases in real estate pricing, ensuring fairness, transparency, and responsible AI use.
9. **Collaboration and Knowledge Sharing**: Foster collaboration with industry stakeholders and academia to drive innovation and knowledge sharing in real estate analytics and AI.

With these advancements, the project can provide more accurate predictions, empower stakeholders with actionable insights, and drive positive transformations in the real estate industry. 