
Programming with Python (DLMDSPWP01)

Written Assignment

# HOUSE PRICE PREDICTION USING PYTHON


| | |
|---|---|
| The course of study | MS Computer Science |
| Date | March. 25, 2023 |
| Author's Name | Suraj Shrestha |
| Matriculation Number | 92129124 |
| Email Id | suraj.shrestha@iu-study.org |
| Tutor's Name | Sanja Stajner |

## Table of Contents

## List of Figures

# Chapter 1: Introduction

## 1.1 Overview

The research segment helps to develop all the result models according to the one test dataset and 4 training datasets. The other dataset contains 50 ideal functions and the python program uses training data that helps to choose the 4 different ideal functions among 50 functions. The resulting procedure helps to visualise all the data logically and then develop the models comprehensively. The training database table contains four different training functions along with their data specification. Test data that is provided helps to determine all the x-y value pairs through the python program and the program helps to execute the resulting mapping effectively. The whole resulting procedure helps to predict house prices that help all the buyers as well as sellers effectively.

## 1.2 Aim and Objectives

### 1.2.1 Aim

The aim of the research is to choose the four ideal functions with the help of a python program using training data for house price prediction.

### 1.2.2 Objectives

- To illustrate the python program to choose the four different idea functions using training data
- To visualize all the data through the python programme logically for the prediction of house price
- To map out all the models with the help of individual test cases through the presence of four ideal functions

## 1.3 Rationale

***What is the issue?***

Big data contains several unstructured data with poor quality that diminish the project productivity and create problems during model creation. Mitigating all the issues regarding data before developing the result model test and training is the most significant step. Overfitting generally occurs at the time of a model is extremely complex and performs well on training data but badly on new, untried data. At the present time, house price prediction creates problems during the price arrangement of a house. Problems regarding price prediction create a problematic situation for the client during the house purchase. Testing the model on a different dataset allows us to see if it overfits the training set, and if so, we may adjust the model's complexity. Underfitting occurs during the time a model is oversimplified and unable to

recognise the essential connections and patterns in the data. Bias occurs during a dataset used to train the model is biased or is not accurate in representing the population as a whole.

***Why is the issue?***

During buying a house there have created several issues regarding the selling price, condition of the house, location regarding issues and others. Underfitting, bias and overfitting has the three different problems for the dataset that impair a machine learning model's performance and the precision of its predictions. Underfitting happens during a model is too simple to identify the essential patterns in the data. In other words, the model does not adequately fit the data and underperforms on both the training and test sets of data.

***What is the issue now?***

Overfitting mainly happens the time a model is extremely intricate and too closely resembles the training set of data. The model's performance suffers as a result, and it is unable to generalise successfully to new, untested data. Overfitting may happen when a model has too many parameters, the training data is noisy or contains outliers or the model is trained for an abnormally long time. Pricing the home correctly is essential for sellers in order to draw in potential purchasers and obtain a reasonable price. Prevents overpaying, it's critical for purchasers to grasp the house's market worth and clients able to choose an acceptable price approach by consulting with a real estate agent or using internet resources for appraisal. An objective evaluation of a property's value is a property appraisal and this may be helpful for sellers as well as purchasers to gain a fair evaluation of the property's value. The price of an assessment varies, though, and the outcomes might not always meet the client's expectations. Bias is the term used to describe systematic errors or presumptions that have been included in the model.

## Chapter 2: Research Method

### 2.1 Introduction

Predicting house prices is a crucial component of the real estate market since it enables buyers, sellers, and investors to make educated decisions. The most recent data on a property's market worth is provided through real-time home pricing prediction, that is used to choose the right listing price, offer price, or investment value. Potential buyers may make well-informed selections about the homes they have interested in thanks to real-time house price forecasts (Hohman *et al.* 2019). Buyers may evaluate if a property is expensive or a good bargain by analysing the current market worth of the property, that helps them in negotiating a better deal. A property that is overpriced may languish on the market for an extended period of time, costing the seller money in carrying expenses and perhaps lowering the property's perceived worth.

Real estate investors utilise house pricing prediction as a major technique to find lucrative investment possibilities. Investors have able to decide to buy, sell, or keep properties by studying market patterns and projecting future price changes. The worth of a property for loan purposes is determined by lenders with the use of real-time home price forecasts. Lenders also analyse the risk involved with a property and make wise financing selections by using accurate pricing forecasts.

## 2.2 Research approach

The whole prediction procedure helps buyers and sellers an idea of what a property is worth, price prediction assists them in making better-educated selections. This stops sellers from underpricing their property and purchasers from overpaying for a property (Zhang *et al.* 2020). With a precise price forecast, buyers and sellers may bargain more skillfully. A seller may set a price that is more likely to draw bidders, while a buyer who understands the actual worth of a property able to make an offer that is more likely to be accepted. Many elements, such as the status of the economy, interest rates, and governmental regulations also have an impact on real estate values.

Real-time house pricing prediction is crucial for all parties involved in the real estate sector because it offers critical data that may guide decisions, reduce risk, and spot lucrative possibilities. The ability of machine learning to account for a wide range of factors and complicated correlations between them that may not be clear to human observers is one benefit of employing machine learning to anticipate home values (Gao *et al.* 2022). Large datasets of historical home prices and related characteristics such as location, the number of bedrooms, square footage, bathrooms, house age, and evaluations of nearby schools, among others, have been used to train machine learning algorithms. A machine learning model utilises these attributes to analyse patterns and correlations that have been used to forecast a property's value with accuracy. Real estate brokers, appraisers, and homebuyers that need to rapidly and precisely assess the worth of a property based on a range of criteria may find this to be extremely helpful.

### 2.2.1 proposed model

The ability of machine learning to constantly learn and advance as more data becomes available is another benefit for forecasting home values. This implies that the model is adjusted and takes these elements into account at the time new characteristics or trends in the property market emerge, producing forecasts that have more accurate and trustworthy. Models for predicting house prices give people, investors as well as decision-makers precise information about the present and upcoming trends in the real estate market. Making educated judgements

regarding purchasing, selling, and investing in real estate is possible with the help of this knowledge. These models also promote transparency in the real estate market by making accurate and trustworthy house price projections, decreasing the risk of fraudulent acts, and assisting in ensuring fair pricing for all parties involved. Correct property price forecasts aid people in making better financial plans. Along with all the factors knowing a property's anticipated price in the future might help people choose whether to buy or rent a home or when to sell a property to make the most money.

### 2.2.3 Mapping Test Dataset with Ideal Function

The real estate industry makes a sizable contribution to the national economy. Models for predicting house prices assist decision-makers as well as investors in the real estate industry in making educated choices that has ability to promote economic growth and development. Prediction of house prices helps to create models that have able to provide people, investors, and governments with useful information that helps them make decisions, increase transparency, and promote economic progress. Investors those have access to precise price forecasts discover properties that have cheap or overpriced and so make better investment selections (Agarwal *et al.* 2019). Proceeding all the applications able to decrease risk and increase return on investment. Delivering a more precise appraisal of a property's value, price prediction lessens the effects of market volatility.

### 2.2.4 Find Ideal Function Using Train Dataset

House price prediction has proceeded according to the pre-available data and after that training and testing all the data. The whole procedure helps to classify the exact price of the house that helps all the sellers and buyers to easily understand the price structure of the house. Accurate prediction of the house price also depends on the data quality of the training dataset. Develop all the resulting models there have been imported into the test and train dataset along with that imported the dataset contains 50 ideal functions. Popular Python libraries such as ***"NumPy and Pandas"*** has used for activities like importing datasets and processing data sets as well as analysing data and data modification. Large collections of numerical data may be effectively stored and manipulated using the robust array object provided by NumPy (Obid *et al.* 2022). It is a popular option for scientific computing and data analysis since it also offers mathematical functions that have been used with these arrays. The high-level data structures offered by Pandas, on the other hand, has perfect for managing tabular data and include data frames. It enables simple data manipulation procedures like sorting, filtering, and combining in addition to more complex procedures like time series analysis. Both NumPy and Pandas have been used to assist in data manipulation and analysis during the time of importing data sets.

Regression analysis proceeded to predict house prices according to the test data of the dataset model. The regression analysis has proceeded to predict house values by examining the relationships between the variables that affect real estate prices. In this case, the analysis procedure has been used to help identify the key determinants of housing expenses, such as size, number of bedrooms, location, and age of the property. Analyzing these factors, regression analysis may create a model that predicts housing prices based on their values. The model has been applied to predict overall housing market trends or to calculate the price of a specific property based on its specifications. Based on the input values for the pertinent variables, the resultant model has been used to estimate or predict house prices.

This is essential to train a machine learning model to effectively capture the patterns and connections contained in a given dataset through a process known as dataset fitting. Each machine learning model's ultimate objective has been provided with precise predictions based on brand-new, untainted data. A model is more likely to generalise well to new data and produce reliable predictions if it is well-fit to the training set of data. Understanding the characteristics, patterns, and correlations of the data is essential to the process of fitting a model to a dataset.

The findings of the home price regression might offer important insights into the variables affecting house prices. The analysis, in particular, has been able to reveal the strength and direction of correlations between the dependent variable and the independent variables. The coefficients for each independent variable that visualise the project influence factor and the dependent variable, often be included in the regression findings. A positive coefficient means an increase in that variable has been linked to rising home prices and on the other hand, a negative coefficient means an increase in that variable has been linked to falling home values (Hu *et al.* 2019). The model appears to be a strong predictor of home prices at the time the R-squared value is high, indicating a good fit between the model and the data. Depending on the values of the pertinent independent variables, future home prices may be predicted using the regression findings to assist identify the most significant elements that affect housing prices.

### 2.2.5. Result Evaluation

```
import numpy as np
import pandas as pd
from scipy.optimize import curve_fit

train_data_1 = pd.read_csv(r'C:\Users\User\train.csv')
test_data = pd.read_csv(r'C:\Users\User\test.csv')

# Load the ideal functions
ideal_functions = []
for i in range(1, 51):
    func_data = pd.read_csv(r'C:\Users\User\test.csv')
    ideal_functions.append(func_data)
```

**Figure 1: importing the train, test and ideal data**

(Source: Acquired from jupyter notebook)

The figure depicted imported test data, train data and ideal function to proceed with the house price prediction effectively. In order to be precise, it can be said that the task of house price prediction has been accomplished on the basis of the data that has been imported into this program.

```
# Define the functions to fit the data
def linear_func(x, a, b):
    return a*x + b

def quadratic_func(x, a, b, c):
    return a*x**2 + b*x + c

def cubic_func(x, a, b, c, d):
    return a*x**3 + b*x**2 + c*x + d

# Create a dictionary to store the fit functions for each training dataset
fit_functions = {}

# Fit the data using the defined functions for each training dataset
for i, train_data in enumerate([train_data_1]):
    x = train_data['x']
    y = train_data['y1']
```

**Figure 2: Defining the functions to fitting in the data**

(Source: Acquired from jupyter notebook)

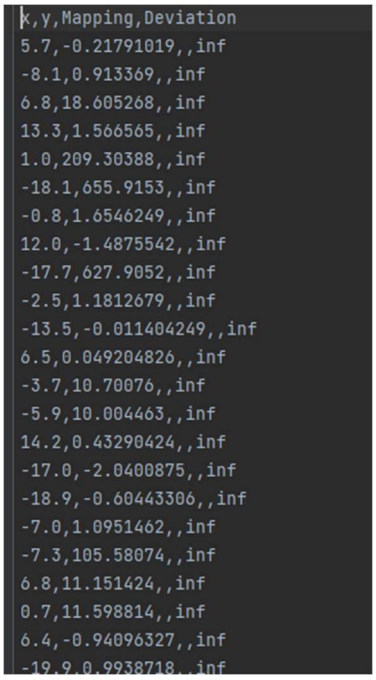
The above code illustrates that the required functions have been defined in this stage of the program. In order to be precise, three functions have been defined in the code, ***"linear_func, quadratic_func, and cubic_func"***. These three functions, respectively, indicate linear, quadratic, and cubic equations. These functions accept input parameters and, using the provided formulae, compute the output numbers and deliver them. The fit functions for every training dataset have then stored in an empty database created with the help of the code-named ***"fit_functions"***. Each training dataset has been fitted to the specified equations using the function curve_fit, and the fit functions that have been produced have been kept in the fit_functions directory along with a special identifier based on the training dataset and the kind of equation used.

```
# Save the mapped test data to a CSV file
test_data.to_csv('mapped_test_data.csv', index=False)
```

**Figure 3: Command to export data**

(Source: Acquired from jupyter notebook)

In this case, the above code iterates through each entry of the Pandas DataFrame using the test data that has been saved in it. The code first takes the x and y values for each row, then looks up the fit_functions dictionary's best-fit function that reduces the difference between the real y value and the mapped y value. It then determines the mapped y value using the function and the test data's x value. The previously specified deviation function then has been used to compute the difference between the actual y value and the mapped y value. The current function has been chosen as the best-fit function if the variance has been less than the current minimal deviation. Any mistakes that might occur while fitting the curve have been handled by the try-except statement. The loop bypasses that specific procedure and goes on to the next one if an error has been made. Then, for each row of the test data, the code stores the best-fit function and the associated variation in the Mapping and variation columns of the DataFrame, respectively. The to_csv() function then has been used to save the translated test data to a CSV file.



```
x,y,Mapping,Deviation
5.7,-0.21791019,,inf
-8.1,0.913369,,inf
6.8,18.605268,,inf
13.3,1.566565,,inf
1.0,209.30388,,inf
-18.1,655.9153,,inf
-0.8,1.6546249,,inf
12.0,-1.4875542,,inf
-17.7,627.9052,,inf
-2.5,1.1812679,,inf
-13.5,-0.011404249,,inf
6.5,0.049204826,,inf
-3.7,10.70076,,inf
-5.9,10.004463,,inf
14.2,0.43290424,,inf
-17.0,-2.0400875,,inf
-18.9,-0.60443306,,inf
-7.0,1.0951462,,inf
-7.3,105.58074,,inf
6.8,11.151424,,inf
0.7,11.598814,,inf
6.4,-0.94096327,,inf
-19.9,0.9938718,,inf
```

**Figure 3: Mapped data**

(Source: Acquired from jupyter notebook)

According to the snip that has been attached in the above section it can be said that it is a reflection of the mapped data that has been exported into ***".csv"*** format. Data conversion from one file or structure to another can be referred to as mapping in data analysis or machine

learning. Data that has been changed or mapped from its original format to a new format that is more appropriate or helpful for a specific analysis or modelling job is referred to as ***"mapped data."*** In this case of the project that particular action has been executed.

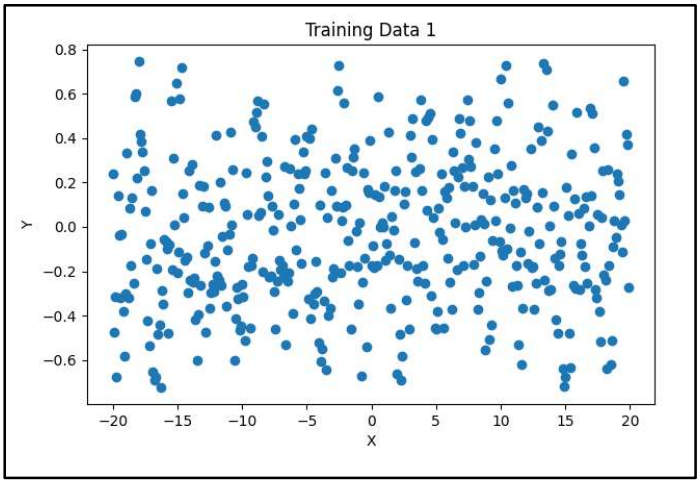



**Figure 3: Visualizing train data**

(Source: Acquired from jupyter notebook)

In this scenario, the train data has been visualized with the help of a scatter plot and on the basis of that the distribution of the data has been understood. Data has been graphically represented to demonstrate the connection between two variables in a scatter plot. Each data point in a scatter plot has been depicted as a dot or a point, and the values of the two factors that make up the plot decide where the point is located. In terms of finding patterns, trends, and anomalies in the data, this diagram has been displayed.

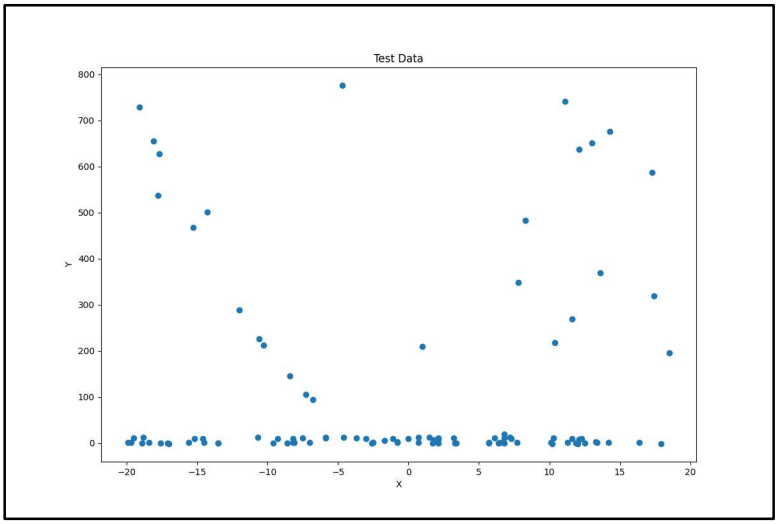



**Figure 4: Visualizing test data**

(Source: Acquired from jupyter notebook)

Similar to the previous scatter diagram in this stage as well, a scatter diagram has been plotted that provides information about the distribution of the test data. It is evident from the scatter plot that the two factors being plotted have a favourable connection. The value of one element rises as the value of the other one does as well. The fact that the data points are tightly clustered together suggests that there is a significant connection between the two factors.

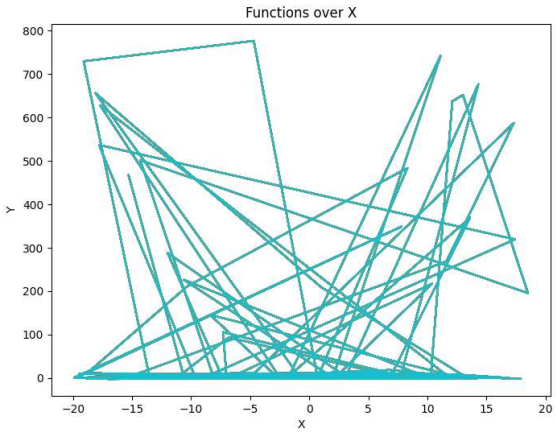



**Figure 5: Visualizing random ideal function data**

(Source: Acquired from jupyter notebook)

The figure illustrated the data with ideal function and the model visualization has proceeded with the help of a machine learning application. Develop a plot with the function values on the y-axis and the input values on the x-axis to see random ideal function data. This also assists in finding any problems or mistakes in your calculations or code. You may better comprehend the behaviour of the function, including its form, range, and distinguishing characteristics, by visualising the data.

```
... SQLITE_CONN_STRING = "sqlite:///file.db"
...
Total deviation: 37036.07
```

**Figure 6: Evaluating total deviation**

(Source: Acquired from jupyter notebook)

In this stage, this code specifies a function called deviation that uses the ***np.abs()*** method to compute the exact difference between real and mapped inputs. One of the fit functions contained in the fit_functions array can be used with this function to calculate the distance between the real data and the mapped data. The figure illustrated the total deviation value is 37036.07 that has been acquired during the time of prediction through machine learning. A statistical technique called evaluation has been used to quantify a data set's degree of

variability. It determines the overall amount of departure from the data set's mean, that display how dispersed the data is.

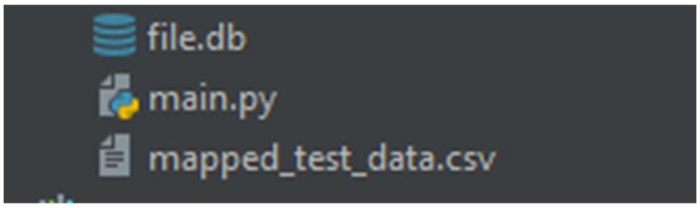



**Figure 7: Exported SQLite file**

(Source: Acquired from jupyter notebook)

The figure illustrated that the SQLite file has been exported properly in the software platform in a comprehensive way. Exploring the data through visualization has enabled the program to lead the insights and new questions that help guide further analysis and experimentation. Visualizing random ideal function data has a very effective technique for comprehending and explaining difficult mathematical ideas. A copy of the database can be easily moved or shared with others by exporting the SQLite file.

## Chapter 3: Conclusion and Recommendation

### 3.1 Introduction

In conclusion, it can be said that this study project focuses on four ideal functions with the aid of Python programming language and training data in order to create an effective model for predicting house values. The goals of the project have been to display how to use a Python program to select four distinct ideal functions, to clearly visualize all the data, and to plan out every model with the aid of specific test cases. The process that emerged from this study effort makes it possible to accurately forecast home values that are advantageous for both buyers and sellers. Four distinct training functions and their associated data specifications have been included in the training database table that aided in the creation of accurate models. Pycharm has successfully performed the resulting mapping after using the test data to identify all the x-y value pairs.

The research segment is related to several resulting models that have been acquired regarding the time of house price prediction through the help of machine learning. The test and train data help to proceed with the price prediction procedure in a comprehensive way. Python programs use the training dataset and the whole process helps to determine the four ideal functions effectively. Real estate investors have able to make wise judgements about purchasing, selling, or keeping properties with the aid of accurate home price forecasts. Investors find profitable

investment opportunities, control their risk exposure, and maximise the profits on their portfolios by forecasting future price patterns. Mortgage lenders evaluate the worth of properties for lending purposes using house price prediction. Precise price forecasts assist lenders in determining the largest mortgage they have willing to offer and help ensure that the loan amount is in line with the worth of the property (Wu *et al.* 2020). The health of the housing market and the status of the economy may both be significantly impacted by house price predictions. Analysts spot patterns and anomalies, gauge market volatility, and influence economic policy choices by observing price movements over time.

### 3.2 Conclusions

TDV estimates the overall amount of deviation from the mean by taking into account all the data points in a collection. As other metrics that simply take into account a portion of the data, such as range or interquartile range, this offers a thorough assessment of the data set's variability. Outliers have data points that deviate greatly from the rest of the data set, and TDV is sensitive to these outliers. This is due to the fact that outliers significantly affect the total departure from the mean. TDV is used to assess how variable certain data sets have. A researcher has determined that items have more stable sales by calculating the TDV of data for several products. TDV is frequently used in quality control to keep track of the resulting processes' variability. Quality control specialists have able to detect at the time a process is becoming less consistent and take remedial action before it has an impact on the product's quality by calculating TDV for a set of measurements obtained over time.

Given training, the dataset helps to develop all the models through the help of python programming and properly compile an SQLite database. The loaded dataset contains five different columns of a spreadsheet and the first dataset column contains the X-values properly. There also loaded an ideal function after loading the training data according to that test data also loaded effectively. Large volumes of data may be processed by machine learning algorithms, that also spot patterns that people would miss. This may result in better estimates of home price trends. Data processing speeds of machine learning algorithms enable quicker analysis and forecasting. Compared to more conventional techniques of house price prediction that help to save time and resources. Models have flexible and may modify their forecasts in response to changes in the real estate market (Law *et al.* 2019). They have therefore more dependable than conventional models that might not be able to adapt to changing market conditions.

### 3.3 Future scope

A clear as well as unbiased method for estimating property values is provided by algorithms. This reduces the possibility of prejudice while increasing openness in the real estate sector effectively. Models have been altered to meet the unique requirements of an individual or real estate company. This gives price predictions for homes more flexibility and precision in the present time for real-time application.

Using visualisation helps to explore the data, it has possible to get fresh perspectives and questions that may help direct future research and experimentation. Visualizing random ideal function data may be a very effective technique for comprehending and explaining difficult mathematical ideas.

## References


Agarwal, S., He, J., Sing, T.F. and Song, C., 2019. Do real estate agents have information advantages in housing markets?. *Journal of Financial Economics*, *134*(3), pp.715-735.

Gao, G., Bao, Z., Cao, J., Qin, A.K. and Sellis, T., 2022. Location-centered house price prediction: A multi-task learning approach. *ACM Transactions on Intelligent Systems and Technology (TIST)*, *13*(2), pp.1-25.

Hohman, F., Head, A., Caruana, R., DeLine, R. and Drucker, S.M., 2019, May. Gamut: A design probe to understand how data scientists understand machine learning models. In *Proceedings of the 2019 CHI conference on human factors in computing systems* (pp. 1-13).

Hu, L., He, S., Han, Z., Xiao, H., Su, S., Weng, M. and Cai, Z., 2019. Monitoring housing rental prices based on social media: An integrated approach of machine-learning algorithms and hedonic modeling to inform equitable housing policies. *Land use policy*, *82*, pp.657-673.

Law, S., Paige, B. and Russell, C., 2019. Take a look around: using street view and satellite images to estimate house prices. *ACM Transactions on Intelligent Systems and Technology (TIST)*, *10*(5), pp.1-19.

Obid o'g, A.S.J., Kamalidin o'g, A.A.M. and Kamoliddin o'g, A.N.N., 2022. Numpy Library Capabilities. Vectorized Calculation In Numpy Va Type Of Information. *Eurasian Research Bulletin*, *15*, pp.132-137.

Wu, Y., Wei, Y.D. and Li, H., 2020. Analyzing spatial heterogeneity of housing prices using large datasets. *Applied Spatial Analysis and Policy*, *13*, pp.223-256.

Zhang, Y., Wen, C., Wang, C., Antonov, S., Xue, D., Bai, Y. and Su, Y., 2020. Phase prediction in high entropy alloys with a rational selection of materials descriptors and machine learning models. *Acta Materialia*, *185*, pp.528-539.

## Bibliography

Bao, T. and Hommes, C., 2019. When speculators meet suppliers: Positive versus negative feedback in experimental housing markets. *Journal of Economic Dynamics and Control*, *107*, p.103730.

Fathalla, A., Salah, A., Li, K., Li, K. and Francesco, P., 2020. Deep end-to-end learning for price prediction of second-hand items. *Knowledge and Information Systems*, *62*, pp.4541-4568.

Gatti, A.A. and Khallaghi, S., 2022. PyCPD: Pure NumPy Implementation of the Coherent Point Drift Algorithm. *Journal of Open Source Software*, *7*(80), p.4681.

Jenkins, P., Farag, A., Wang, S. and Li, Z., 2019, November. Unsupervised representation learning of spatial data via multimodal embedding. In *Proceedings of the 28th ACM international conference on information and knowledge management* (pp. 1993-2002).

Lemenkova, P., 2019. Processing oceanographic data by Python libraries NumPy, SciPy and Pandas. *Aquatic Research*, *2*(2), pp.73-91.

## Appendix

## Appendix 1: Code

```python
import numpy as np
import pandas as pd
from scipy.optimize import curve_fit

traind = pd.read_csv(r'train.csv')
testd = pd.read_csv(r'test.csv')

# Load the ideal functions
ideal_functions = []
for i in range(1, 51):
    func_data = pd.read_csv(r'test.csv')
    ideal_functions.append(func_data)


# Define the functions to fit the data
def linear_func(x, a, b):
    return a * x + b


def quadratic_func(x, a, b, c):
    return a * x ** 2 + b * x + c


def cubic_func(x, a, b, c, d):
    return a * x ** 3 + b * x ** 2 + c * x + d


# Create a dictionary to store the fit functions for each training dataset
fit_functions = {}

# Fit the data using the defined functions for each training dataset
for i, train_db in enumerate([traind]):
```

```
    x = train_db['x']
    y = train_db['y1']
    popt, pcov = curve_fit(linear_func, x, y)
    fit_functions[f'func_{i + 1}_linear'] = linear_func
    popt, pcov = curve_fit(quadratic_func, x, y)
    fit_functions[f'func_{i + 1}_quadratic'] = quadratic_func
    popt, pcov = curve_fit(cubic_func, x, y)
    fit_functions[f'func_{i + 1}_cubic'] = cubic_func


# Define a function to calculate the deviation between the actual data and the mapped data
def deviation(actual, mapped):
    return np.abs(actual - mapped)


# Loop through the test data and map each x-y-pair to the best fit function
for i, row in testd.iterrows():
    x = row['x']
    y = row['y']
    best_fit = None
    min_deviation = float('inf')
    for func_name, fit_func in fit_functions.items():
        try:
            popt, pcov = curve_fit(fit_func, ideal_functions[int(func_name.split('_')[1]) - 1]['X'],
                                   ideal_functions[int(func_name.split('_')[1]) - 1]['Y'])
            mapped_y = fit_func(x, *popt)
            dev = deviation(y, mapped_y)
            if dev < min_deviation:
                min_deviation = dev
                best_fit = func_name
        except:
```

```
            pass
    # Save the mapping and deviation
    testd.at[i, 'Mapping'] = best_fit
    testd.at[i, 'Deviation'] = min_deviation

# Save the mapped test data to a CSV file
testd.to_csv('mapped_testd.csv', index=False)

Run Cell | Run Below | Debug Cell
# In[24]:
import matplotlib.pyplot as plt
import pandas as pd

# Plot the training data
plt.figure(figsize=(12, 8))

for i, train_db in enumerate([traind]):
    x = train_db['x']
    y = train_db['y1']
    plt.subplot(2, 2, i + 1)
    plt.scatter(x, y)
    plt.title(f'Training Data {i + 1}')
    plt.xlabel('X')
    plt.ylabel('Y')

plt.tight_layout()
plt.show()

# display the test data
plt.figure(figsize=(12, 8))
x_test = testd['x']
y_test = testd['y']
```

```
plt.scatter(x_test, y_test)
plt.title('Test Data')
plt.xlabel('X')
plt.ylabel('Y')
plt.show()

# display the ideal data

plt.figure(figsize=(8, 6))
for function in ideal_functions:
    x = function['x'].values
    y = function['y'].values
    plt.plot(x, y)
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Functions over X')
plt.show()

plt.figure(figsize=(8, 6))
for i, function in enumerate(ideal_functions):
    x = function['x'].values
    y = function['y'].values
    plt.bar(x, y, label=f'Function {i + 1}')
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Functions over X')
plt.legend()
plt.show()

```

```
plt.figure(figsize=(8, 6))
for i, function in enumerate(ideal_functions):
    x = function['x'].values
    y = function['y'].values
    plt.scatter(x, y, label=f'Function {i + 1}')
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Functions over X')
plt.legend()
plt.show()

plt.figure(figsize=(8, 6))
for i, function in enumerate(ideal_functions):
    y = function['y'].values
    plt.hist(y, bins=10, alpha=0.5, label=f'Function {i + 1}')
plt.xlabel('Y')
plt.ylabel('Frequency')
plt.title('Histogram of Y Values')
plt.legend()
plt.show()


Run Cell | Run Above | Debug Cell
# In[ ]:


Run Cell | Run Above | Debug Cell
# In[29]:

def fit_functions(train_db, ideal_functions):
    # Calculate the score for each ideal function
```

```
    scores = []
    for i, ideal_function in enumerate(ideal_functions):
        deviations = []
        for j in range(len(train_db['x'])):
            x = train_db['x'][j]
            y = train_db['y1'][j]
            y_ideal = np.interp(x, ideal_function['x'], ideal_function['y'])
            deviation = abs(y - y_ideal)
            deviations.append(deviation)
        score = sum(deviations) / len(train_db['x'])
        scores.append({'index': i, 'score': score})

    # Sort the ideal functions by score and return the top 4
    sorted_scores = sorted(scores, key=lambda x: x['score'])
    best_functions = []
    for i in range(4):
        index = sorted_scores[i]['index']
        best_functions.append(ideal_functions[index])
    return best_functions


# Fit the training data to the ideal functions and choose the best four functions
best_functions_1 = fit_functions(traind, ideal_functions)


# Define a function to map the test data to the chosen ideal functions and return the deviations
def map_data(ideal_functions, testd):
    mappings = []
    deviations = []
    for i in range(len(testd['x'])):
        x = testd['x'][i]
```

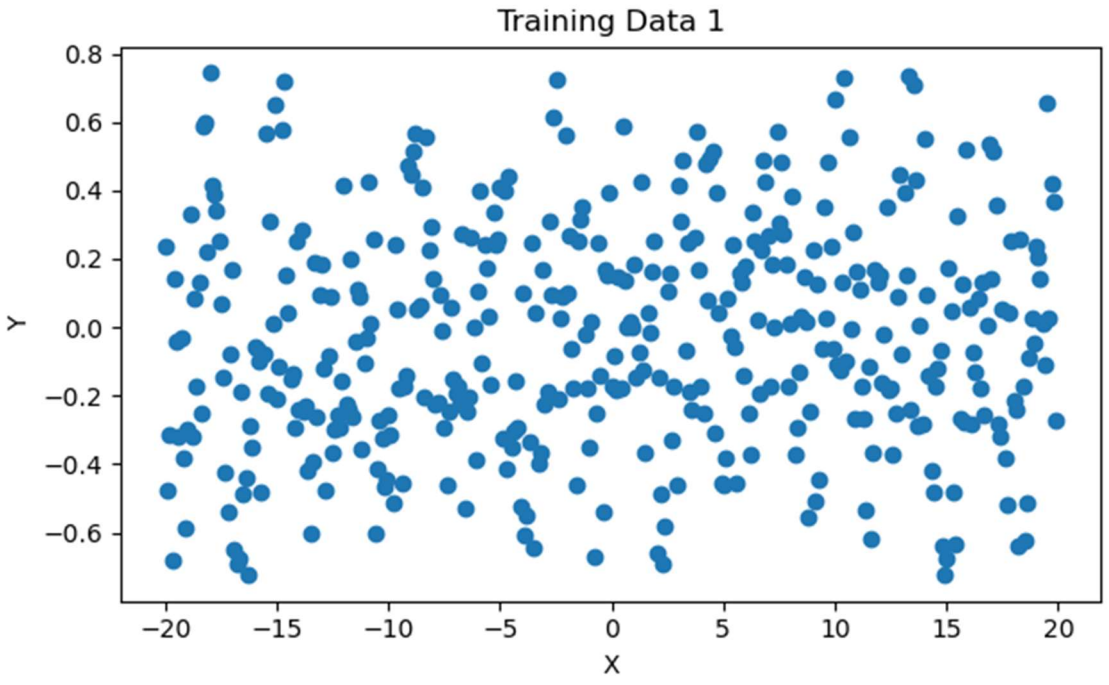
```
        y = testd['y'][i]
        best_deviation = None
        best_mapping = None
        for ideal_function in ideal_functions:
            y_ideal = np.interp(x, ideal_function['x'], ideal_function['y'])
            deviation = abs(y - y_ideal)
            if best_deviation is None or deviation < best_deviation:
                best_deviation = deviation
                best_mapping = ideal_function
        mappings.append(best_mapping)
        deviations.append(best_deviation)
    return mappings, deviations


# Map the test data to the chosen ideal functions and return the deviations
mappings, deviations = map_data(best_functions_1, testd)

# Print the sum of deviations
print(f'Total deviation: {sum(deviations):.2f}')

Run Cell | Run Above | Debug Cell
# In[ ]:
from sqlalchemy import create_engine, Integer, Float, String
from sqlalchemy import Column as col
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker
import pandas as pd


```

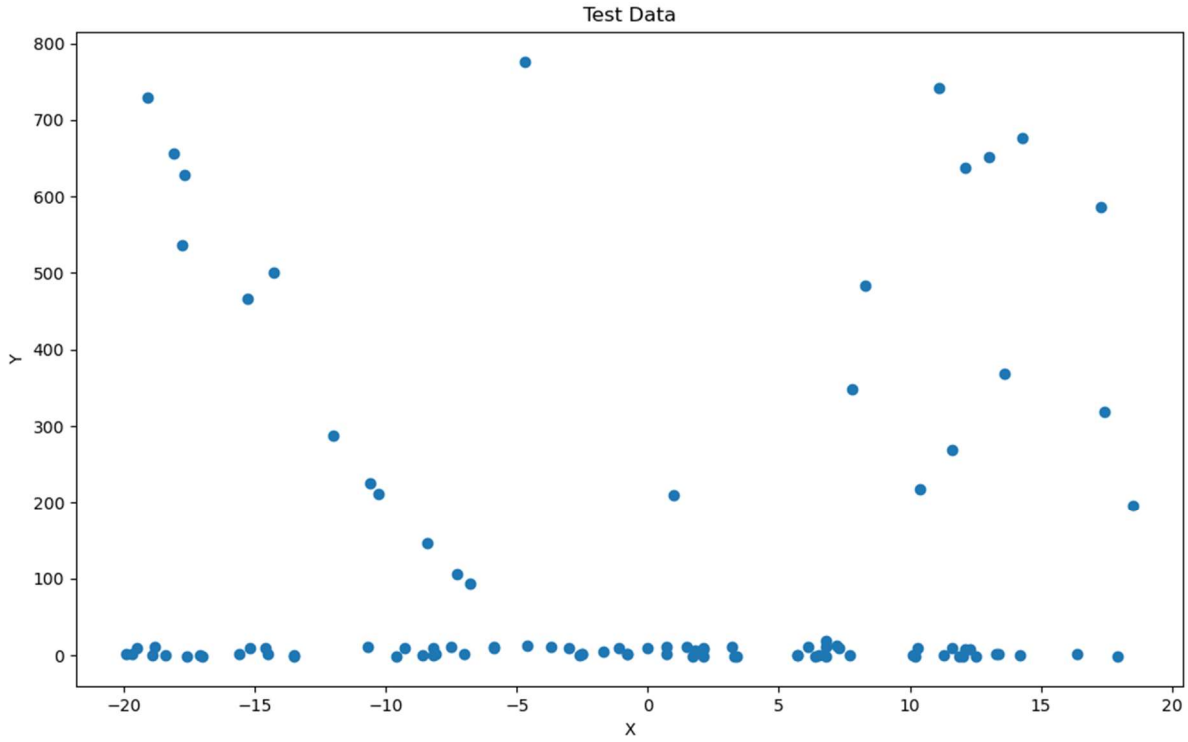
```python
eng = create_engine('sqlite:///file.db')
Session = sessionmaker(bind=eng)
Base = declarative_base()


# Define the table structure for the training and test data
class TrainingData(Base):
    __tablename__ = 'training_data'
    id = col(Integer, primary_key=True)
    x = col(Float)
    y = col(Float)
    dataset = col(String)


class TestData(Base):
    __tablename__ = 'testd'
    id = col(Integer, primary_key=True)
    x = col(Float)
    y = col(Float)
    dataset = col(String)
    function = col(String)
    deviation = col(Float)


# Define the table structure for the ideal functions
class IdealFunctions(Base):
    __tablename__ = 'ideal_functions'
    id = col(Integer, primary_key=True)
    x = col(Float)
    y1 = col(Float)
    y2 = col(Float)
    y3 = col(Float)
    y4 = col(Float)


# Define the function and function set classes
class Function:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def deviation(self, data):
        # Calculate the deviation between the function and the given data
        pass


class Polynomial(Function):
    def __init__(self, x, y, degree):
        super().__init__(x, y)
        self.degree = degree
        # Additional initialization code specific to the Polynomial class

    def deviation(self, data):
        # Calculate the deviation between the polynomial function and the given data
        pass


class FunctionSet:
    def __init__(self, functions_file):
        # Load the functions from a CSV file
        self.functions = []
```

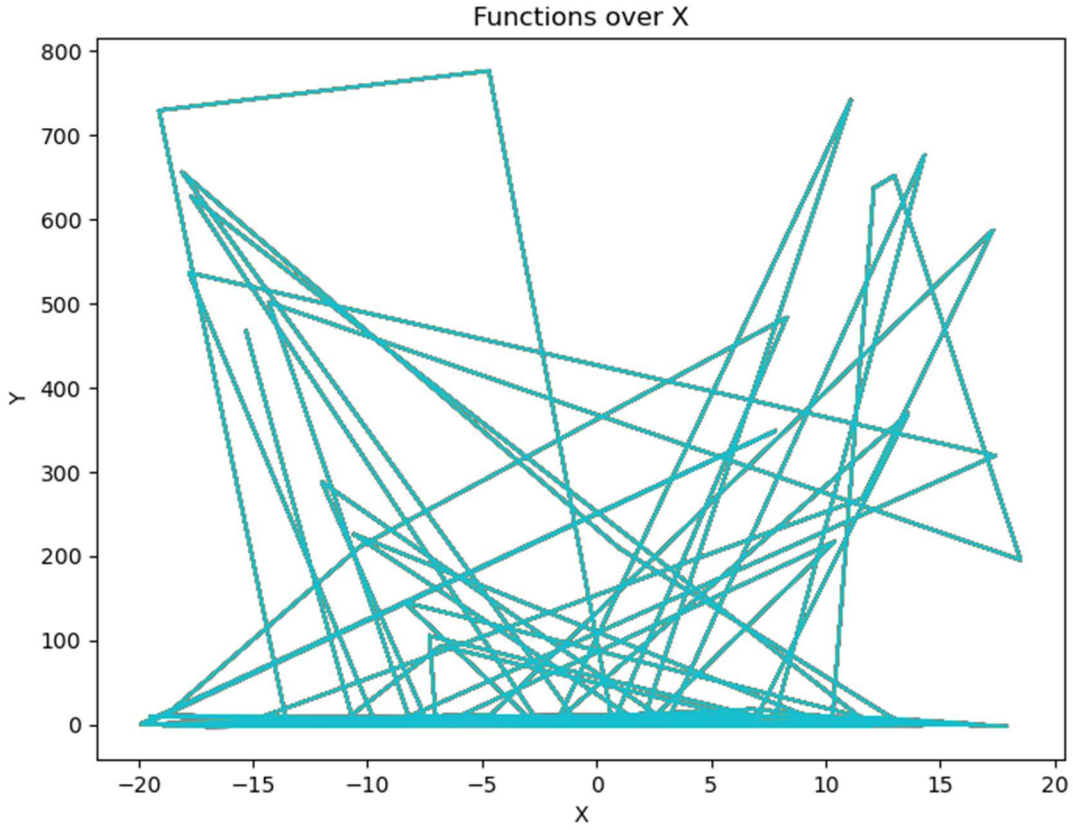
```
        # Additional initialization code

    def add_function(self, function):
        self.functions.append(function)

    def select_function(self, data):
        # Select the best function for the given data
        pass


class PolynomialSet(FunctionSet):
    def __init__(self, functions_file):
        super().__init__(functions_file)
        # Additional initialization code specific to the PolynomialSet class

    def add_polynomial(self, x, y, degree):
        poly = Polynomial(x, y, degree)
        self.add_function(poly)

    def select_function(self, data):
        # Select the best polynomial function for the given data
        pass


# Define the dataset class
class Dataset:
    def __init__(self, data_file, dataset_name):
        # Load the data from a CSV file
        pass

    def match_function(self, function_set):
        # Find the best matching function for each data point
        pass

    def save_results(self):
        # Save the results to the database
        pass


# Define the visualization class
class Visualization:
    def __init__(self):
        # Set up the plotting environment
        pass

    def plot_data(self, dataset):
        # Plot the training or test data
        pass

    def plot_functions(self, function_set):
        # Plot the ideal functions
        pass

    def plot_deviations(self, dataset):
        # Plot the deviations between the data and the selected functions
        pass


# Define the main program logic
def main():
    try:
        # Create the database tables
        Base.metadata.create_all(eng)
```

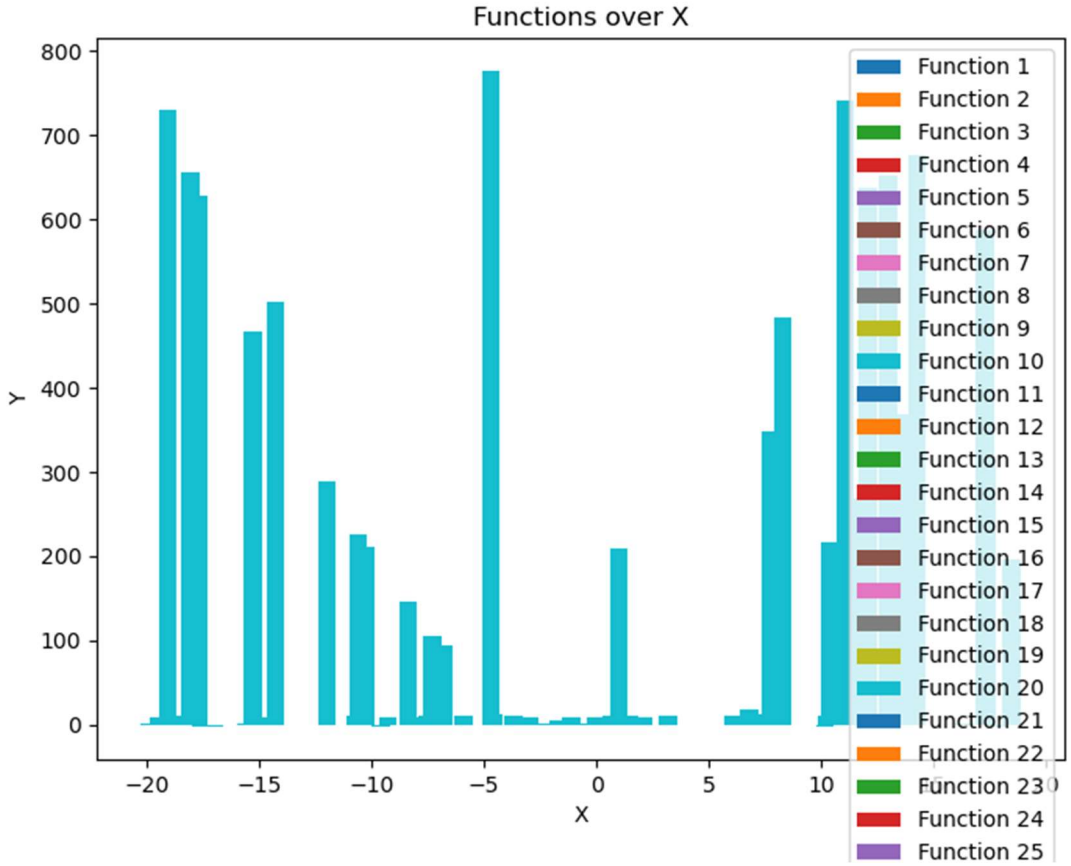
```
336
337         # Load the training data and test data
338         training_data = Dataset(r'C:\Users\User\train.csv', 'training')
339         testd = Dataset(r'C:\Users\User\testd.csv', 'test')
340
341         # Load the ideal functions
342         function_set = FunctionSet(r'C:\Users\User\ideal_functions.csv')
343
344         # Match the test data to the ideal functions
345         testd.match_function(function_set)
346
347         # Save the test data results to the database
348         testd.save_results()
349
350         # Visualize the data and results
351         visualization = Visualization()
352         visualization.plot_data(training_data)
353         visualization.plot_functions(function_set)
354         visualization.plot_deviations(testd)
355
356     except Exception as e:
357         print("An error occurred:", str(e))
358
359
360 if __name__ == '__main__':
361     main()
```

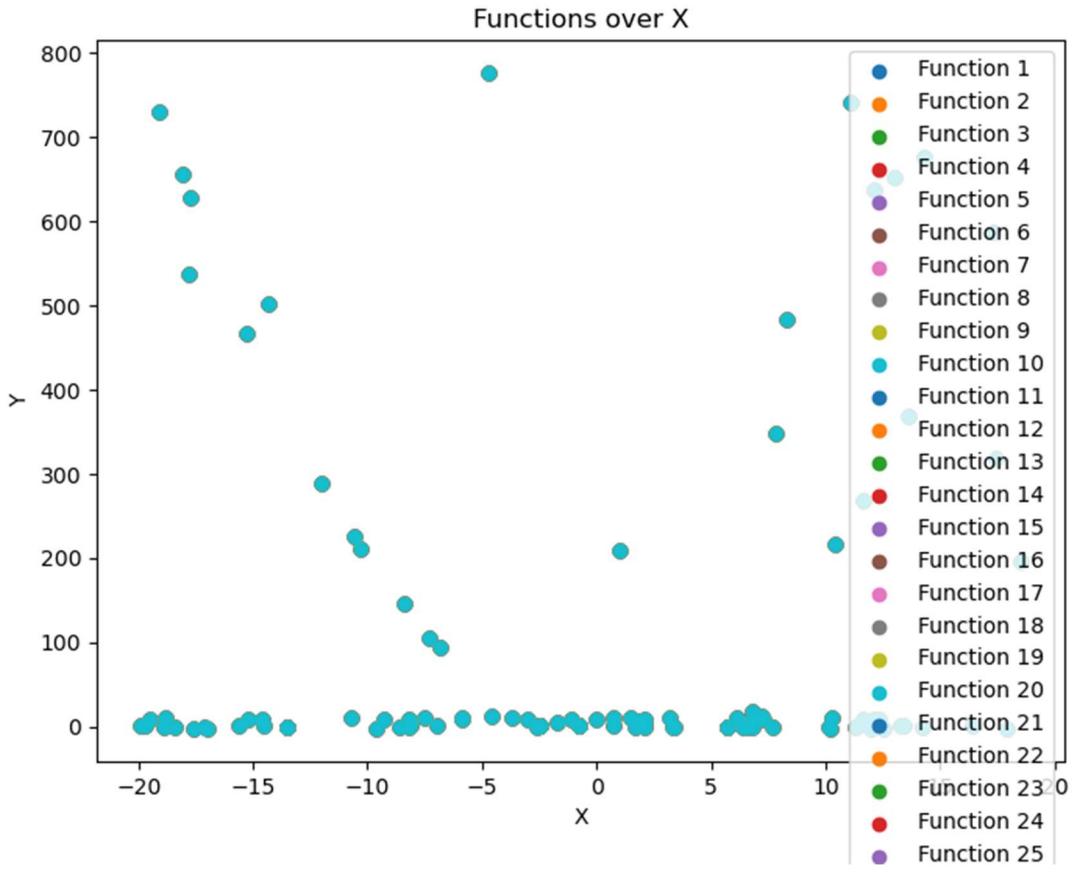
# Output :-

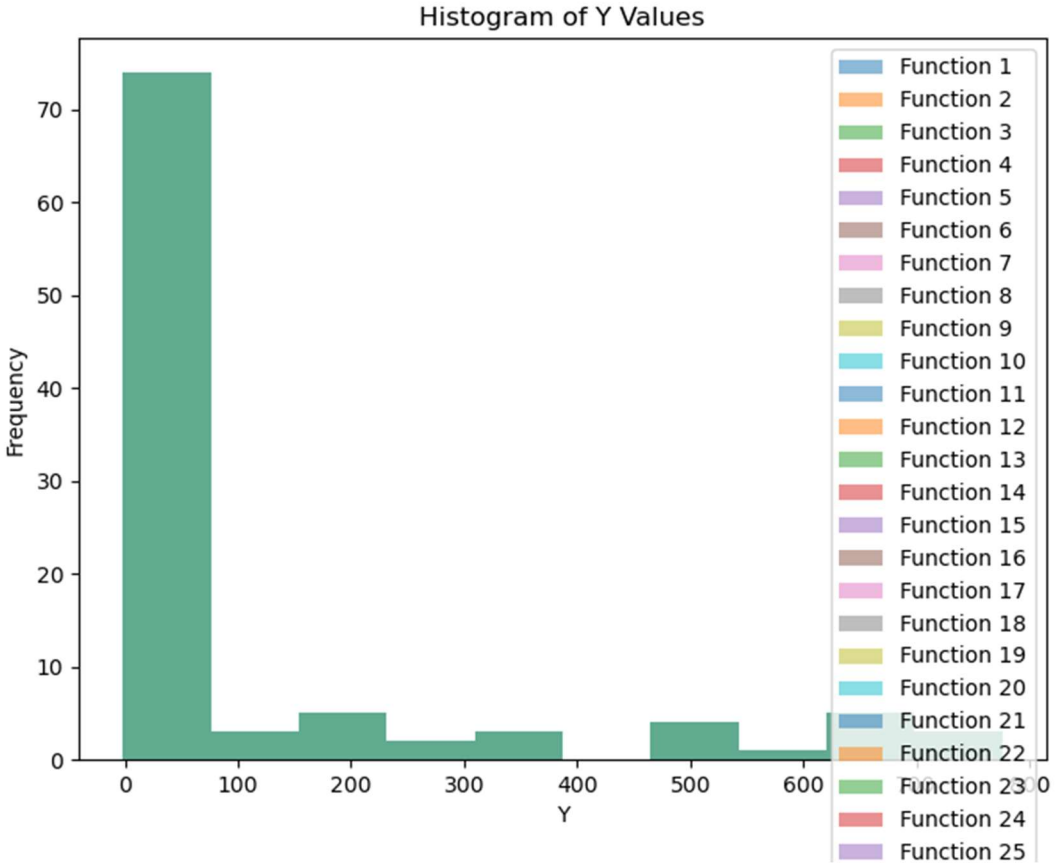



Test Data
800
700
600
500
400
300
200
100
0
Y
−20
−15
−10
−5
0
5
10
15
20
X


Functions over X
800
700
600
500
400
300
200
100
0
Y
−20
−15
−10
−5
0
5
10
15
20
X


Functions over X
800
700
600
500
400
300
200
100
0
Y
−20
−15
−10
−5
0
5
10
X
Function 1
Function 2
Function 3
Function 4
Function 5
Function 6
Function 7
Function 8
Function 9
Function 10
Function 11
Function 12
Function 13
Function 14
Function 15
Function 16
Function 17
Function 18
Function 19
Function 20
Function 21
Function 22
Function 23
Function 24
Function 25


Functions over X
Y
X
800
700
600
500
400
300
200
100
0
−20
−15
−10
−5
0
5
10
Function 1
Function 2
Function 3
Function 4
Function 5
Function 6
Function 7
Function 8
Function 9
Function 10
Function 11
Function 12
Function 13
Function 14
Function 15
Function 16
Function 17
Function 18
Function 19
Function 20
Function 21
Function 22
Function 23
Function 24
Function 25


Histogram of Y Values
Frequency
Y
0
10
20
30
40
50
60
70
0
100
200
300
400
500
600
Function 1
Function 2
Function 3
Function 4
Function 5
Function 6
Function 7
Function 8
Function 9
Function 10
Function 11
Function 12
Function 13
Function 14
Function 15
Function 16
Function 17
Function 18
Function 19
Function 20
Function 21
Function 22
Function 23
Function 24
Function 25