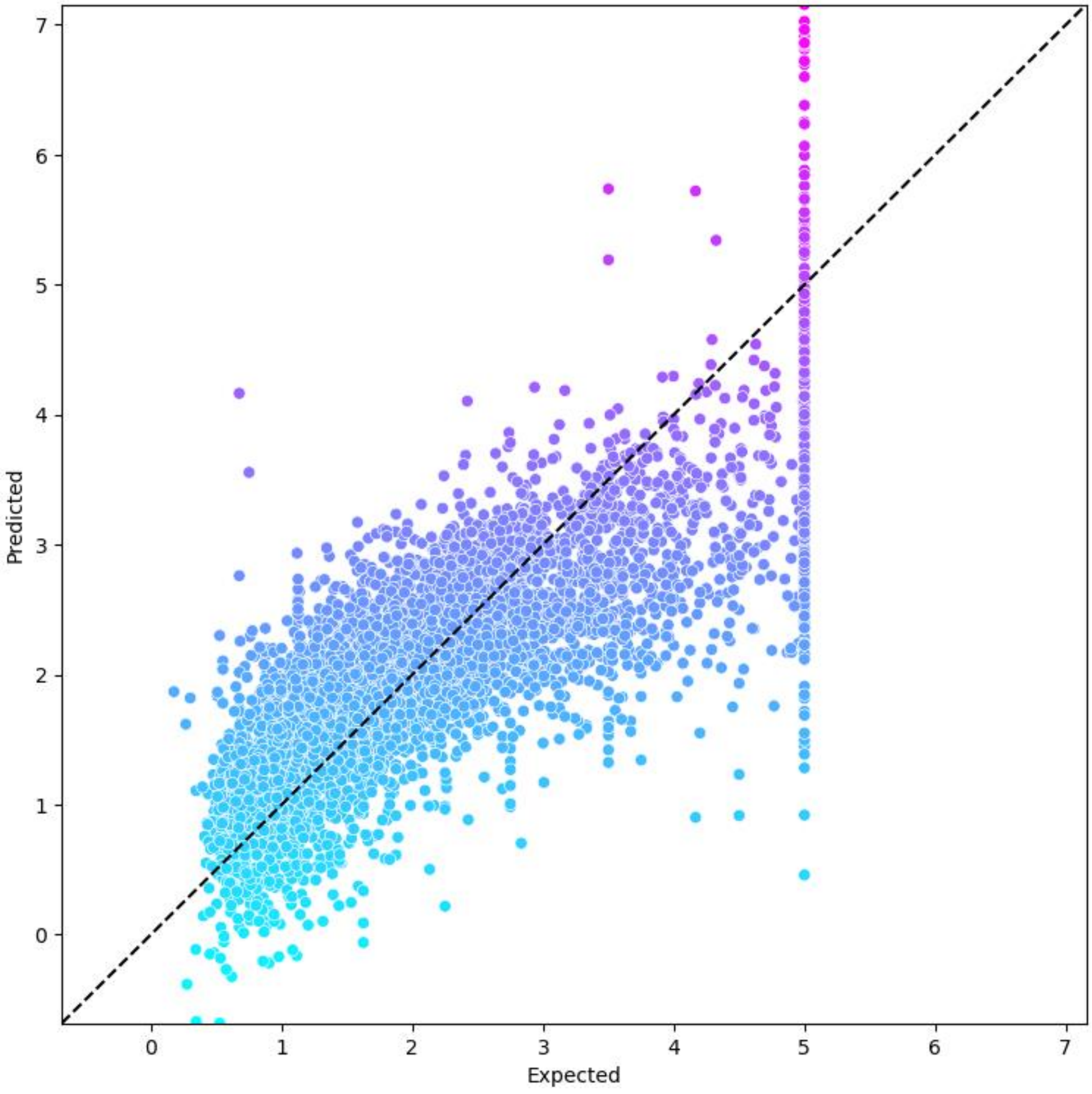
Algonquin College of Applied Arts and Technology

Business Intelligence System Infrastructure

CST 2101 – Business Intelligence Programming Project: Data Analysis



SUBMITTED BY	*STUDENT NUMBER*
ESHA ESHA	**041041612**

Contents

PROBLEM STATEMENT

Business intelligence includes strategies and technologies used by organizations for data analysis, predictions, and management of information. Common functions of business intelligence include predictive and prescriptive analysis as a part of that we have two datasets from the sklearn datasets library toy datasets:

Please find the links to the datasets used:

SKLEARN - REAL WORLD DATASET - California Housing Dataset

SKLEARN TOY DATASETS – Diabetes Dataset

1. California Housing
2. Diabetes

To examine this methodically and provide meaningful analysis we have designed a program that allows user to choose between different options and the ability to restart at anytime:

1. Load the dataset.
2. Explore Dataset
3. Split Dataset
4. Train Dataset
5. Test Dataset
6. Visualise Dataset
7. Regression

SPECIFICATIONS:

Two menus

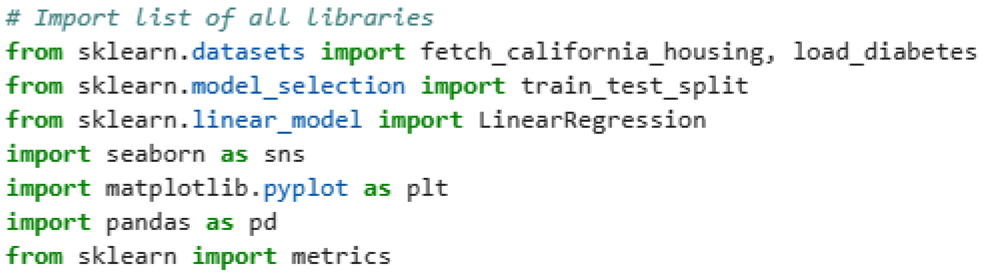
Main menu and Menu no. 2

Menu 1 is choosing between the datasets and existing/ ending from the program

Menu 2 gives user the option for chosen dataset to be loaded/ explored/ spitted/trained/tested/visualise and regress.

APPROACH: THE CODE

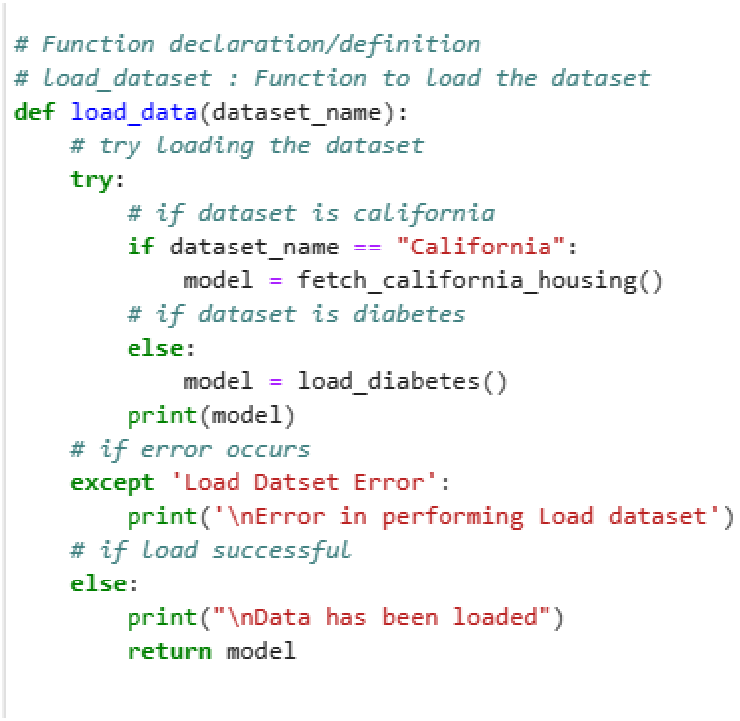
Importing all the necessary libraries.



```
# Import list of all libraries
from sklearn.datasets import fetch_california_housing, load_diabetes
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
from sklearn import metrics
```

Snippet 1

- **Loading the dataset choosing from two datasets: California and diabetes defining the functions :**



```
# Function declaration/definition
# load_dataset : Function to load the dataset
def load_data(dataset_name):
    # try loading the dataset
    try:
        # if dataset is california
        if dataset_name == "California":
            model = fetch_california_housing()
        # if dataset is diabetes
        else:
            model = load_diabetes()
        print(model)
    # if error occurs
    except 'Load Datset Error':
        print('\nError in performing Load dataset')
    # if load successful
    else:
        print("\nData has been loaded")
        return model
```

Snippet 2

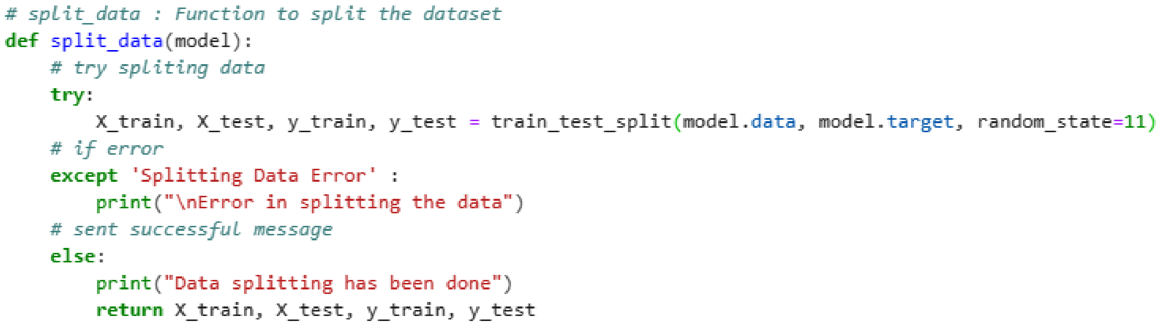
1. dataset_name= “California”

model= fetch_calfornia_housing()

3. If it’s diabetes:
   model = load_diabetes()

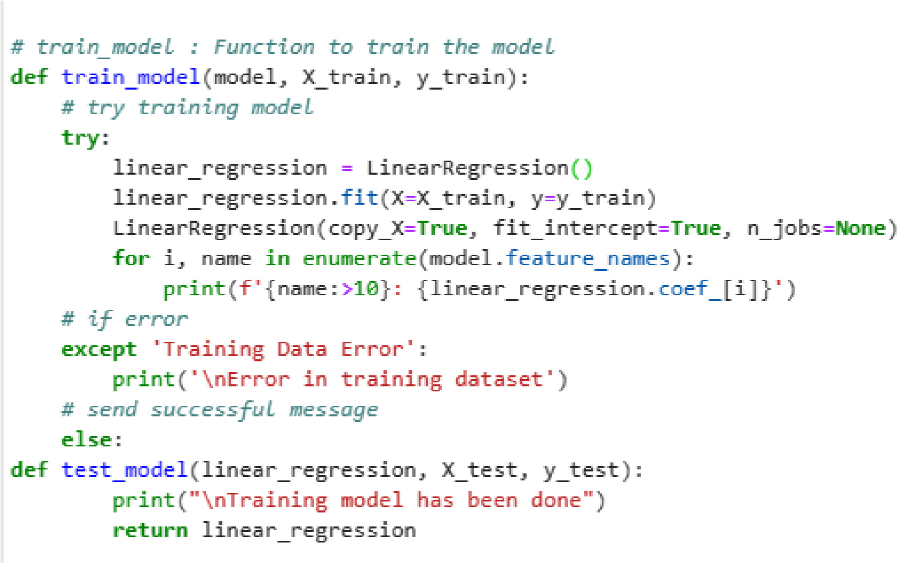
- **Exploring the data.**

- **Splitting the data**

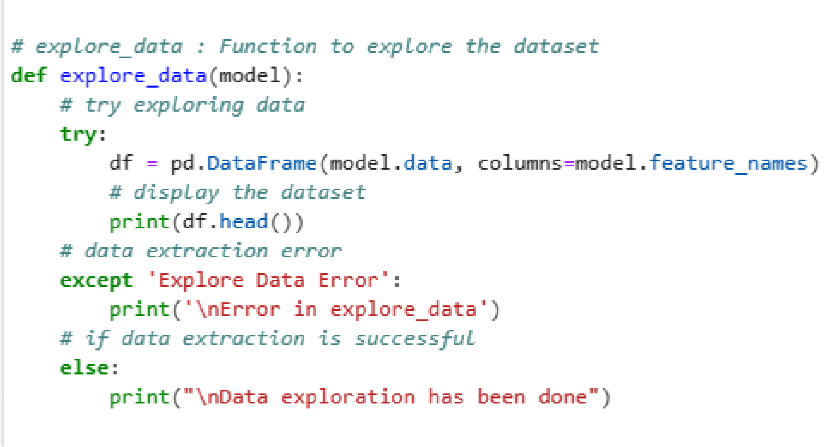


```
# split_data : Function to split the dataset
def split_data(model):
    # try spliting data
    try:
        X_train, X_test, y_train, y_test = train_test_split(model.data, model.target, random_state=11)
    # if error
    except 'Splitting Data Error' :
        print("\nError in splitting the data")
    # sent successful message
    else:
        print("Data splitting has been done")
        return X_train, X_test, y_train, y_test
```

- **Training the model**

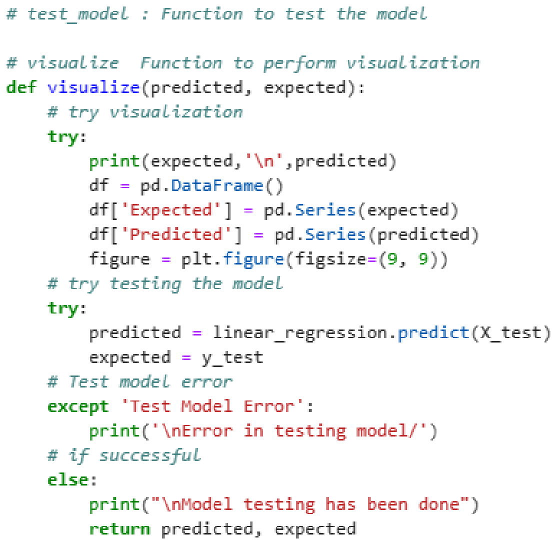


```
# train_model : Function to train the model
def train_model(model, X_train, y_train):
    # try training model
    try:
        linear_regression = LinearRegression()
        linear_regression.fit(X=X_train, y=y_train)
        LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None)
        for i, name in enumerate(model.feature_names):
            print(f'{name:>10}: {linear_regression.coef_[i]}')
    # if error
    except 'Training Data Error':
        print('\nError in training dataset')
    # send successful message
    else:
def test_model(linear_regression, X_test, y_test):
        print("\nTraining model has been done")
        return linear_regression
```



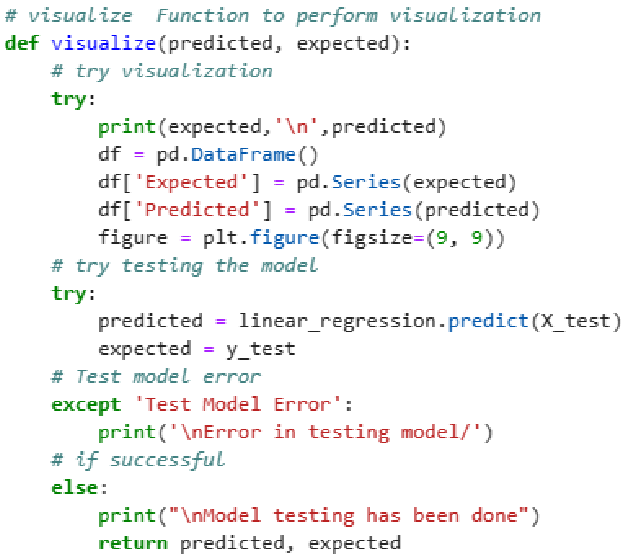
```
# explore_data : Function to explore the dataset
def explore_data(model):
    # try exploring data
    try:
        df = pd.DataFrame(model.data, columns=model.feature_names)
        # display the dataset
        print(df.head())
    # data extraction error
    except 'Explore Data Error':
        print('\nError in explore_data')
    # if data extraction is successful
    else:
        print("\nData exploration has been done")
```

➢ Testing the model



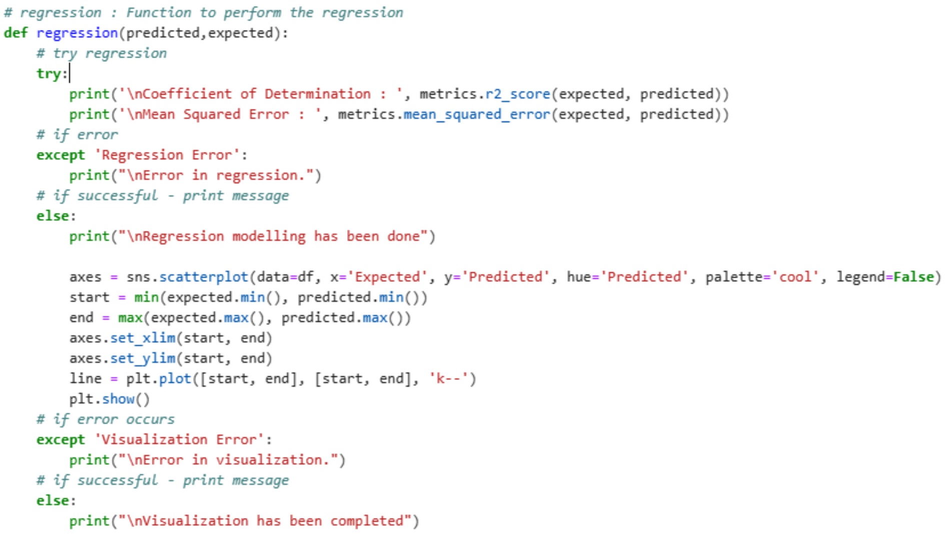
```
# test_model : Function to test the model

# visualize  Function to perform visualization
def visualize(predicted, expected):
    # try visualization
    try:
        print(expected,'\n',predicted)
        df = pd.DataFrame()
        df['Expected'] = pd.Series(expected)
        df['Predicted'] = pd.Series(predicted)
        figure = plt.figure(figsize=(9, 9))
    # try testing the model
    try:
        predicted = linear_regression.predict(X_test)
        expected = y_test
    # Test model error
    except 'Test Model Error':
        print('\nError in testing model/')
    # if successful
    else:
        print("\nModel testing has been done")
        return predicted, expected
```

➢ Visualizing the model



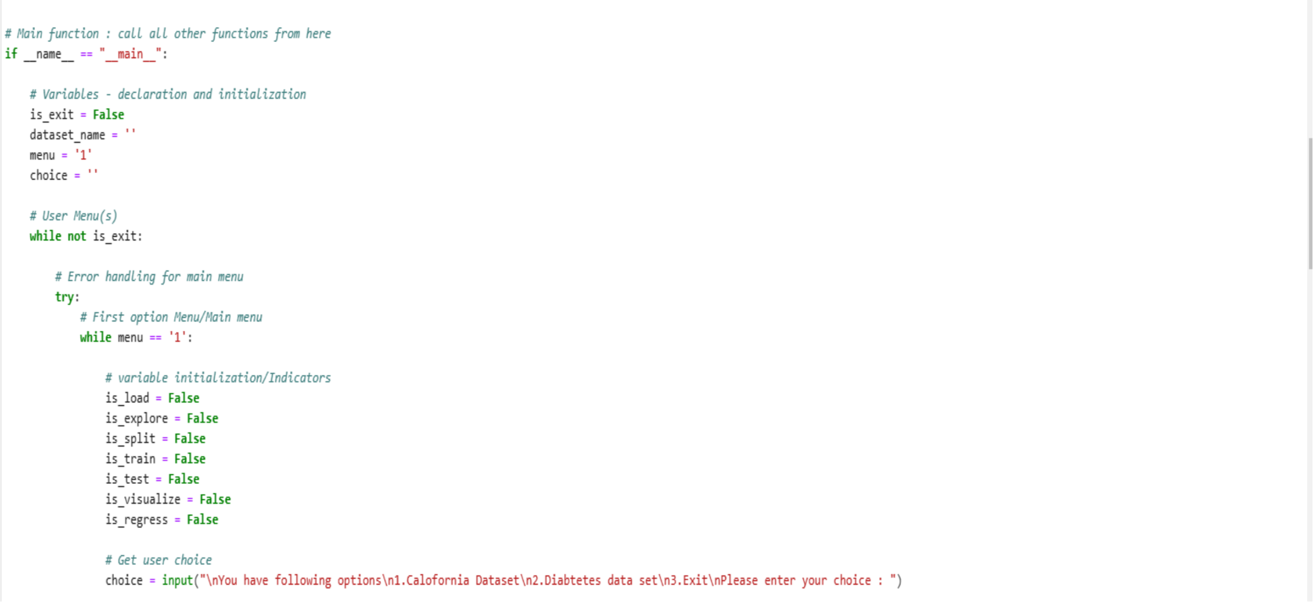
```
# visualize  Function to perform visualization
def visualize(predicted, expected):
    # try visualization
    try:
        print(expected,'\n',predicted)
        df = pd.DataFrame()
        df['Expected'] = pd.Series(expected)
        df['Predicted'] = pd.Series(predicted)
        figure = plt.figure(figsize=(9, 9))
    # try testing the model
    try:
        predicted = linear_regression.predict(X_test)
        expected = y_test
    # Test model error
    except 'Test Model Error':
        print('\nError in testing model/')
    # if successful
    else:
        print("\nModel testing has been done")
        return predicted, expected
```

- **Regression model.**



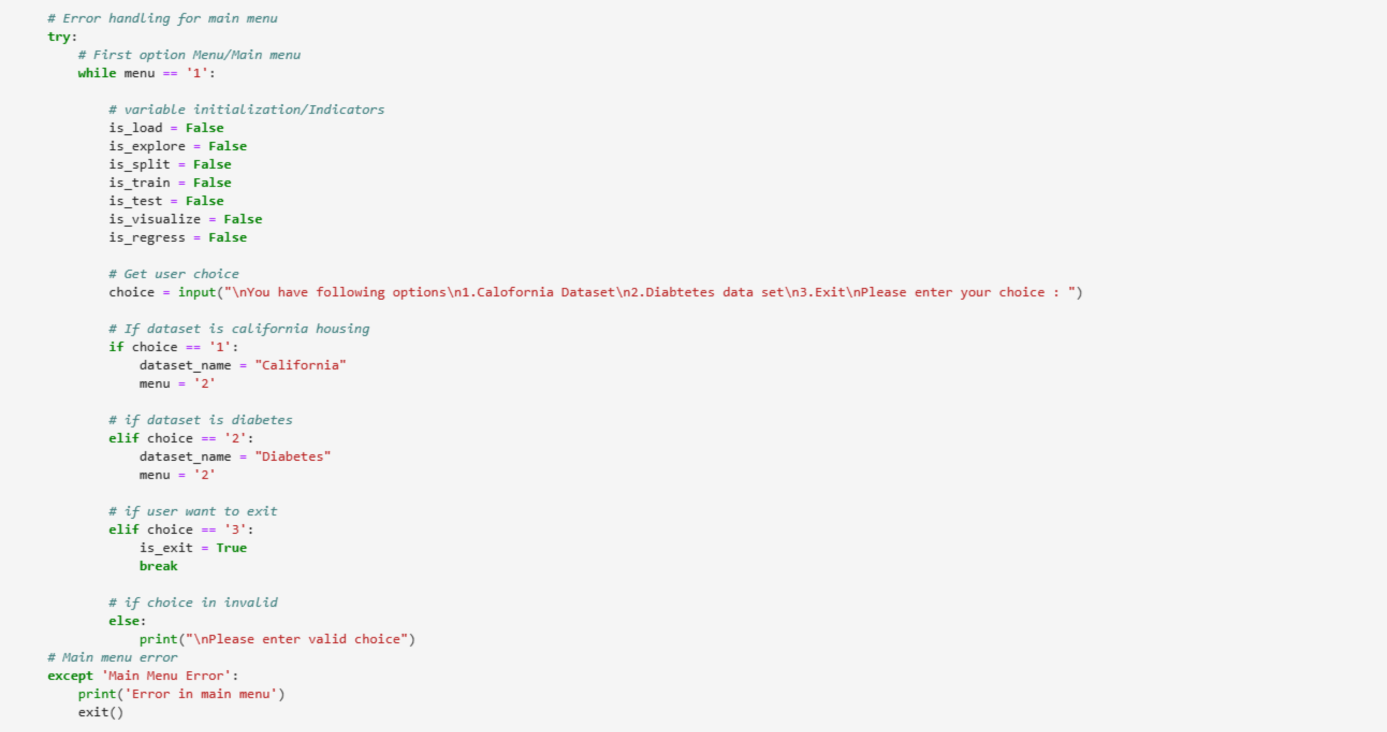
```
# regression : Function to perform the regression
def regression(predicted,expected):
    # try regression
    try:
        print('\nCoefficient of Determination : ', metrics.r2_score(expected, predicted))
        print('\nMean Squared Error : ', metrics.mean_squared_error(expected, predicted))
    # if error
    except 'Regression Error':
        print("\nError in regression.")
    # if successful - print message
    else:
        print("\nRegression modelling has been done")

        axes = sns.scatterplot(data=df, x='Expected', y='Predicted', hue='Predicted', palette='cool', legend=False)
        start = min(expected.min(), predicted.min())
        end = max(expected.max(), predicted.max())
        axes.set_xlim(start, end)
        axes.set_ylim(start, end)
        line = plt.plot([start, end], [start, end], 'k--')
        plt.show()
    # if error occurs
    except 'Visualization Error':
        print("\nError in visualization.")
    # if successful - print message
    else:
        print("\nVisualization has been completed")
```

- **Calling the main function and providing user the option to choose from:**



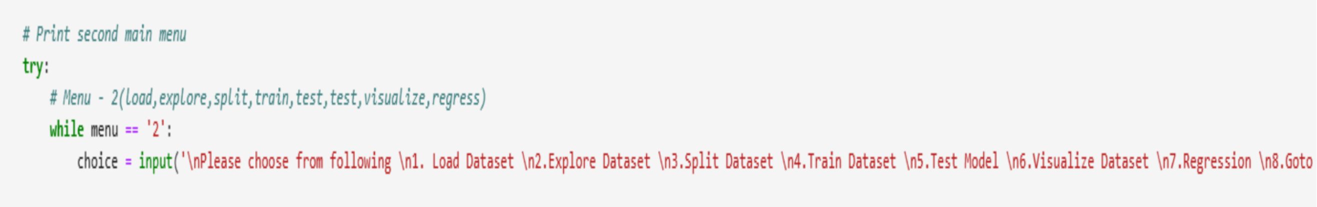
```
# Main function : call all other functions from here
if __name__ == "__main__":

    # Variables - declaration and initialization
    is_exit = False
    dataset_name = ''
    menu = '1'
    choice = ''

    # User Menu(s)
    while not is_exit:

        # Error handling for main menu
        try:
            # First option Menu/Main menu
            while menu == '1':

                # variable initialization/Indicators
                is_load = False
                is_explore = False
                is_split = False
                is_train = False
                is_test = False
                is_visualize = False
                is_regress = False

                # Get user choice
                choice = input("\nYou have following options\n1.Calofornia Dataset\n2.Diabtetes data set\n3.Exit\nPlease enter your choice : ")
```

- **Error Handling for the main menu:**



```python
# Error handling for main menu
try:
    # First option Menu/Main menu
    while menu == '1':

        # variable initialization/Indicators
        is_load = False
        is_explore = False
        is_split = False
        is_train = False
        is_test = False
        is_visualize = False
        is_regress = False

        # Get user choice
        choice = input("\nYou have following options\n1.Calofornia Dataset\n2.Diabtetes data set\n3.Exit\nPlease enter your choice : ")

        # If dataset is california housing
        if choice == '1':
            dataset_name = "California"
            menu = '2'

        # if dataset is diabetes
        elif choice == '2':
            dataset_name = "Diabetes"
            menu = '2'

        # if user want to exit
        elif choice == '3':
            is_exit = True
            break

        # if choice in invalid
        else:
            print("\nPlease enter valid choice")
# Main menu error
except 'Main Menu Error':
    print('Error in main menu')
    exit()
```

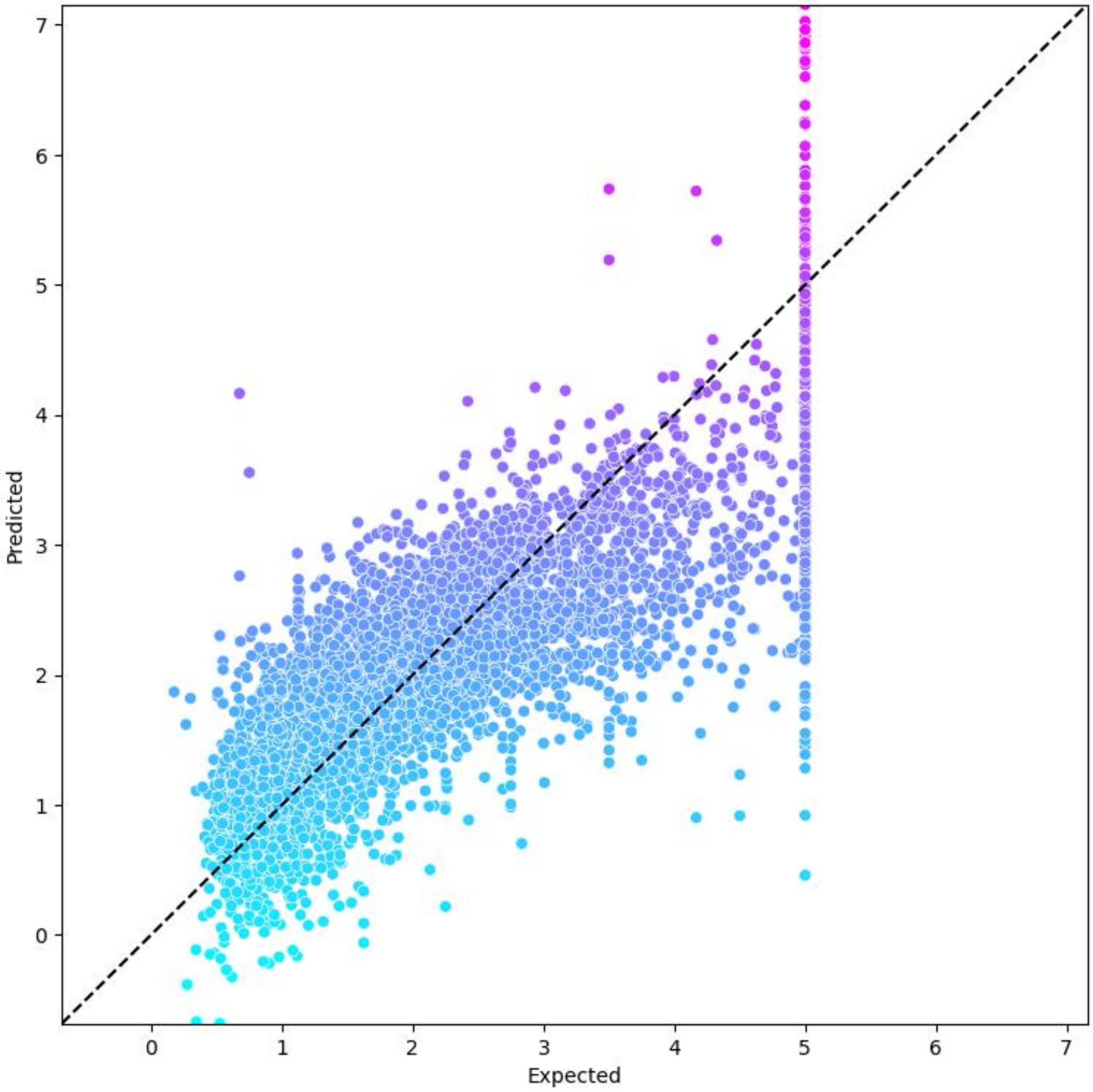
- **Second main menu option:**



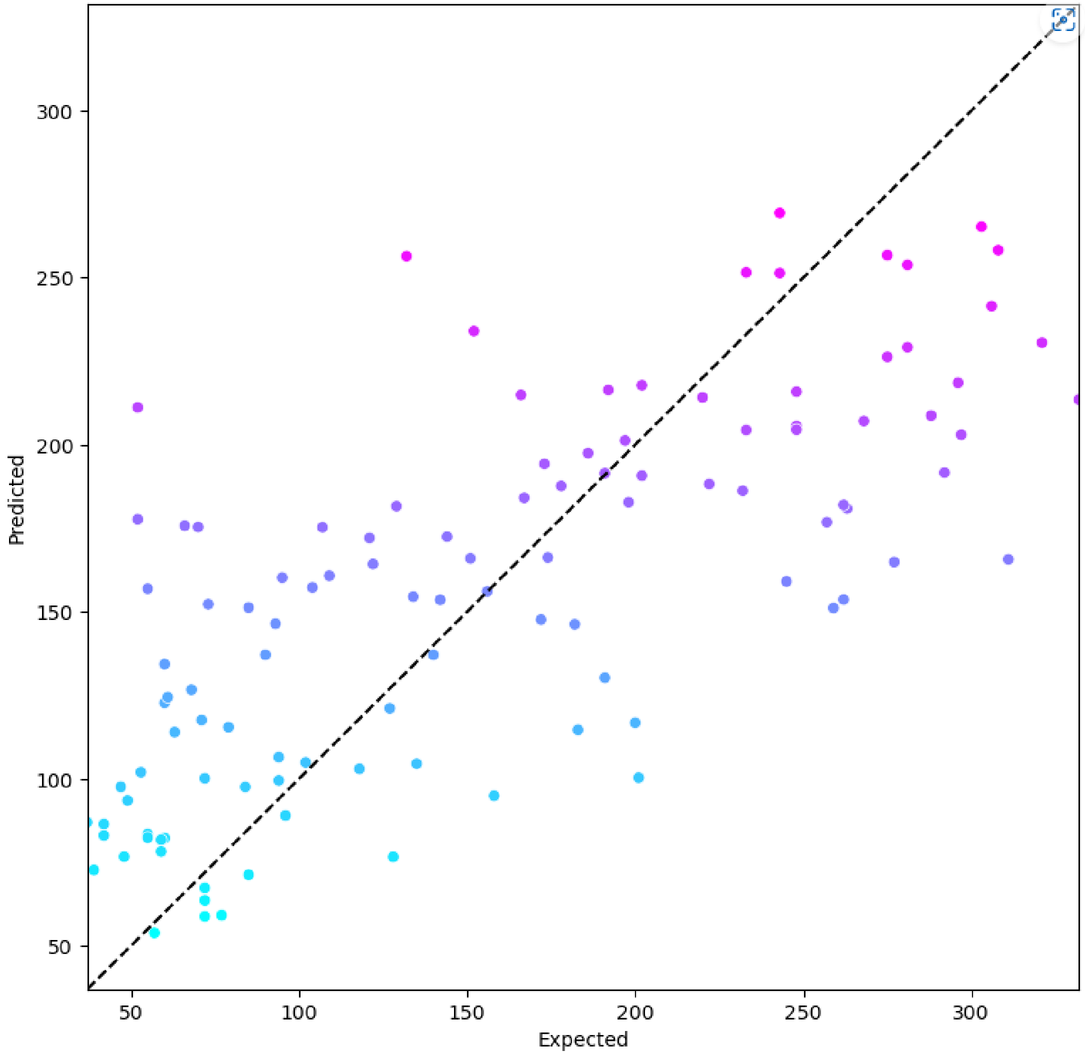
```python
# Print second main menu
try:
    # Menu - 2(load,explore,split,train,test,test,visualize,regress)
    while menu == '2':
        choice = input('\nPlease choose from following \n1. Load Dataset \n2.Explore Dataset \n3.Split Dataset \n4.Train Dataset \n5.Test Model \n6.Visualize Dataset \n7.Regression \n8.Goto
```

- **Visualizing the model:**

1. **California dataset depicting expected v/s predicted.**



2.Diabetes depicting expected v/s predicted



DESIGN:

The solution is designed keeping *modularity* in mind. The solution is based on the functionality. The system has many functionalities and hence it is difficult to put pseudo code of all functionalities.

SYSTEM DESIGN DIAGRAM:

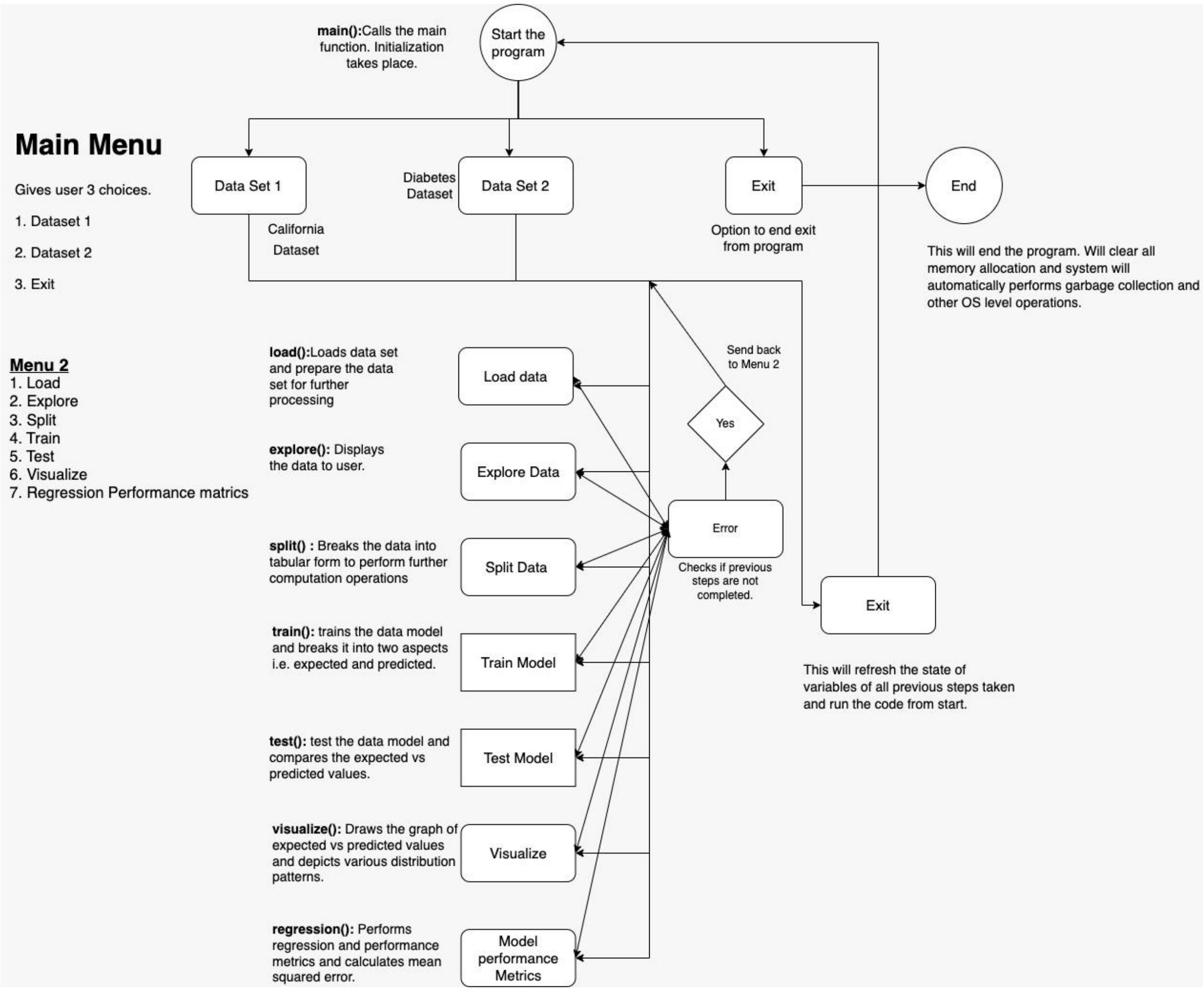

main():Calls the main function. Initialization takes place.
Start the program
Main Menu
Gives user 3 choices.
1. Dataset 1
2. Dataset 2
3. Exit
Data Set 1
California Dataset
Diabetes Dataset
Data Set 2
Exit
Option to end exit from program
End
This will end the program. Will clear all memory allocation and system will automatically performs garbage collection and other OS level operations.
Menu 2
1. Load
2. Explore
3. Split
4. Train
5. Test
6. Visualize
7. Regression Performance matrics
load():Loads data set and prepare the data set for further processing
Load data
explore(): Displays the data to user.
Explore Data
split() : Breaks the data into tabular form to perform further computation operations
Split Data
train(): trains the data model and breaks it into two aspects i.e. expected and predicted.
Train Model
test(): test the data model and compares the expected vs predicted values.
Test Model
visualize(): Draws the graph of expected vs predicted values and depicts various distribution patterns.
Visualize
regression(): Performs regression and performance metrics and calculates mean squared error.
Model performance Metrics
Send back to Menu 2
Yes
Error
Checks if previous steps are not completed.
Exit
This will refresh the state of variables of all previous steps taken and run the code from start.

Figure 1 Logical Design Flowchart

In the above diagram:
Main Menu:
Giving user 3 choices:

1. **Dataset 1**
2. **Dataset 2**
3. **Exit**

Menu 2:
1.Loading the dataset.

2.Exploring the dataset.
3. Splitting the dataset.
4. Training
5. Testing
6. Visualizing
7. Regression

TEST PLAN

This report comes along with a testing plan that is included in the format provided. Please refer to the .xlsx file in this submission to view the test plan.

REFRENCES:

1. https://www.w3schools.com/.
2. https://learning.oreilly.com/library/view/intro-to-python/9780135404799/xhtml/fileP7001016447000000000000000006B2F.xhtml#P70010164470000000000000000006B2F.