Configuration Manual

MSc Research Project
Data Analytics

Chandrashekar Gettam Rajgopal
Student ID: X21226075

School of Computing
National College of Ireland

Supervisor: Mayank Jain

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Chandrashekar Gettam Rajgopal
Student ID:	X21226075
Programme:	Data Analytics
Year:	2023
Module:	MSc Research Project
Supervisor:	Mayank Jain
Submission Due Date:	14/12/2023
Project Title:	Configuration Manual
Word Count:	1468
Page Count:	10

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Chandrashekar Gettam Rajgopal
Date:	29th January 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☑
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☑
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☑

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Chandrashekar Gettam Rajgopal
X21226075

1 Introduction

This configuration manual contains step-by-step instructions for configuring the system and the manual's main goal is to provide comprehensive instructions on how to carry out the research study. It also details the machine setup needed to compile and execute the models, model evaluations, code snippets, and visualizations from exploratory data research. The processes involve installing the necessary programs and packages in addition to the minimal configuration that is advised for a project to succeed.

2 System Configuration

2.1 Hardware Requirements

- Operating System: Google Colab will run on a modern web browser and all Operating systems that can run a modern web browser can be used.
- RAM: Minimum 4GB but recommended 8GB for better performance.
- Processor: Modern processor Intel, AMD or Apple Silicon which can handle multiple tasks and run multiple browser tabs smoothly.
- Storage: Minimum of 256 GB Hard disk or Solid state drive to install the browser and to run.

2.2 Software Requirements

- Web Browser: Mozilla Firefox, Google Chrome, Safari, or any other Chromium browser for a good experience.
- Google Drive: Store the dataset in Google Drive and mount it to Google Colab.
- Microsoft Excel: To do initial exploration.
- Tableau: Used for initial exploration and exploratory data analysis.
- Python: Any version of Python 3.x

3 Code snippets

3.1 Data Collection

Data is downloaded from Kaggle using this link Investments VC Dataset Arindam235 (2023). The dataset contains an investments_VC.csv file that has information on various companies investment details.

3.2 Setup Google Colab

Store the downloaded investments_VC.csv file in Google Drive and mount the Google Drive into Google Colab as shown in Figure 1.

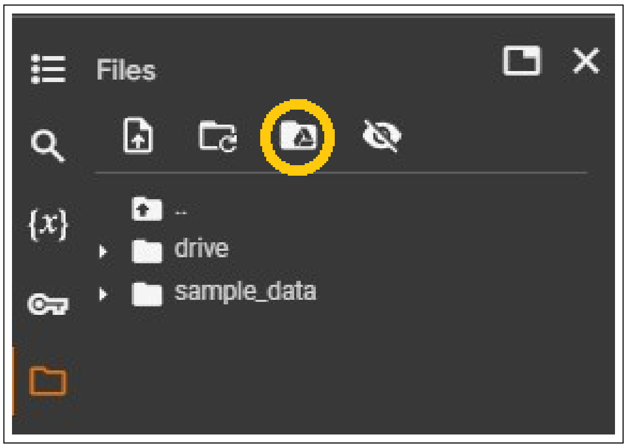
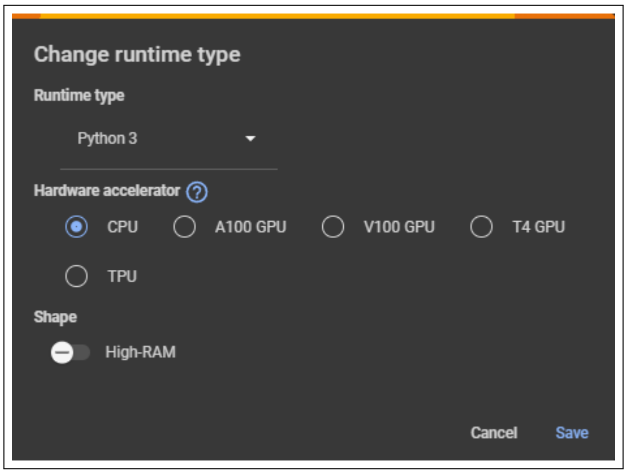


Figure 1: Google Drive mount in Google Colab

Choosing the run time of Google Colab Figure 2. Choose Python for Runtime type and CPU can be chosen by default and click on save.



Change runtime type

Runtime type

Python 3

Hardware accelerator

● CPU ○ A100 GPU ○ V100 GPU ○ T4 GPU

○ TPU

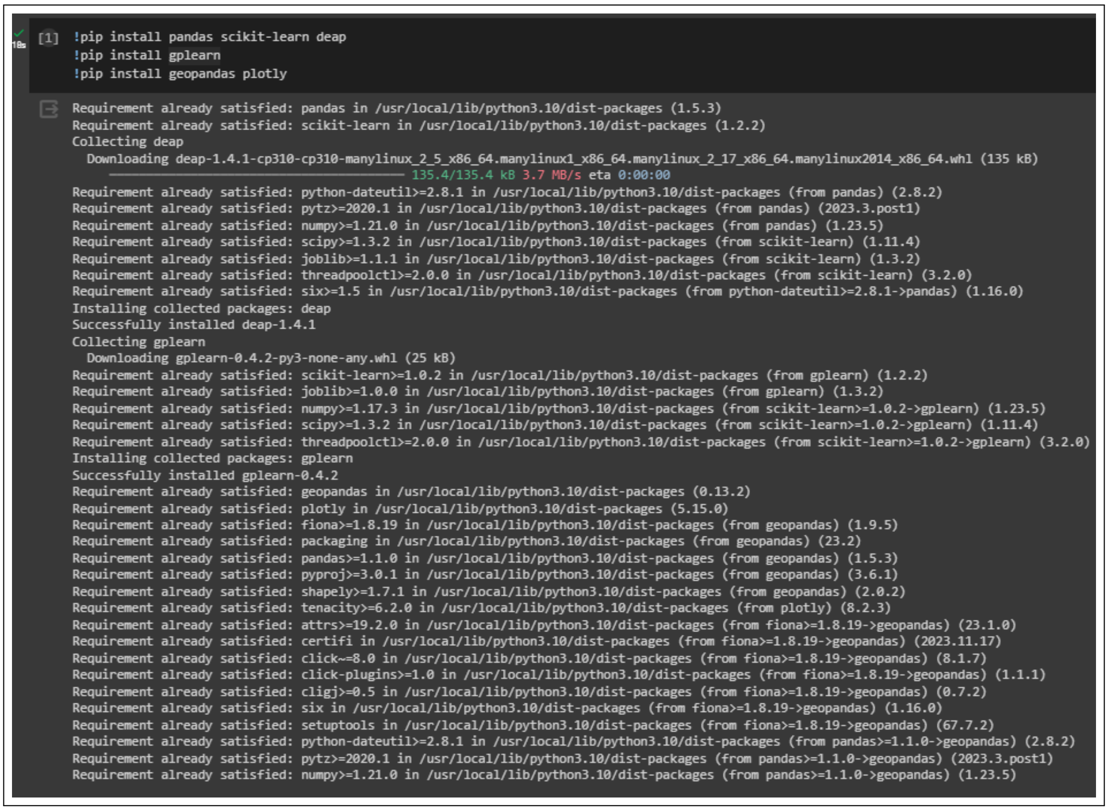
Shape

High-RAM

Figure 2: Google Colab Runtime

3.3 Install necessary Packages and Libraries

The next step is installing Pandas, Scikit-learn, Deap, gplearn, geopandas and plotly. packages as shown in Figure 3.



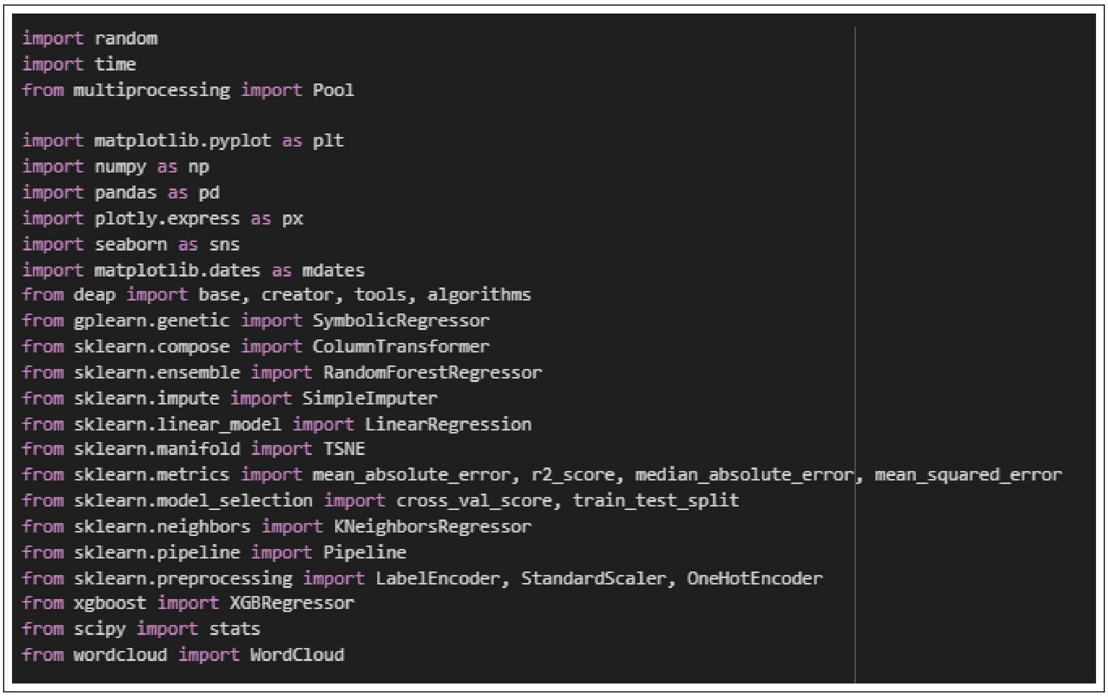
```
[1] !pip install pandas scikit-learn deap
    !pip install gplearn
    !pip install geopandas plotly
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (1.5.3)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (1.2.2)
Collecting deap
  Downloading deap-1.4.1-cp310-cp310-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manylinux2014_x86_64.whl (135 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 135.4/135.4 kB 3.7 MB/s eta 0:00:00
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2023.3.post1)
Requirement already satisfied: numpy>=1.21.0 in /usr/local/lib/python3.10/dist-packages (from pandas) (1.23.5)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.11.4)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (3.2.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.1->pandas) (1.16.0)
Installing collected packages: deap
Successfully installed deap-1.4.1
Collecting gplearn
  Downloading gplearn-0.4.2-py3-none-any.whl (25 kB)
Requirement already satisfied: scikit-learn>=1.0.2 in /usr/local/lib/python3.10/dist-packages (from gplearn) (1.2.2)
Requirement already satisfied: joblib>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from gplearn) (1.3.2)
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=1.0.2->gplearn) (1.23.5)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=1.0.2->gplearn) (1.11.4)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=1.0.2->gplearn) (3.2.0)
Installing collected packages: gplearn
Successfully installed gplearn-0.4.2
Requirement already satisfied: geopandas in /usr/local/lib/python3.10/dist-packages (0.13.2)
Requirement already satisfied: plotly in /usr/local/lib/python3.10/dist-packages (5.15.0)
Requirement already satisfied: fiona>=1.8.19 in /usr/local/lib/python3.10/dist-packages (from geopandas) (1.9.5)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from geopandas) (23.2)
Requirement already satisfied: pandas>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from geopandas) (1.5.3)
Requirement already satisfied: pyproj>=3.0.1 in /usr/local/lib/python3.10/dist-packages (from geopandas) (3.6.1)
Requirement already satisfied: shapely>=1.7.1 in /usr/local/lib/python3.10/dist-packages (from geopandas) (2.0.2)
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from plotly) (8.2.3)
Requirement already satisfied: attrs>=19.2.0 in /usr/local/lib/python3.10/dist-packages (from fiona>=1.8.19->geopandas) (23.1.0)
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-packages (from fiona>=1.8.19->geopandas) (2023.11.17)
Requirement already satisfied: click~=8.0 in /usr/local/lib/python3.10/dist-packages (from fiona>=1.8.19->geopandas) (8.1.7)
Requirement already satisfied: click-plugins>=1.0 in /usr/local/lib/python3.10/dist-packages (from fiona>=1.8.19->geopandas) (1.1.1)
Requirement already satisfied: cligj>=0.5 in /usr/local/lib/python3.10/dist-packages (from fiona>=1.8.19->geopandas) (0.7.2)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from fiona>=1.8.19->geopandas) (1.16.0)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from fiona>=1.8.19->geopandas) (67.7.2)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.1.0->geopandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.1.0->geopandas) (2023.3.post1)
Requirement already satisfied: numpy>=1.21.0 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.1.0->geopandas) (1.23.5)
```

Figure 3: Packages

Next step is importing all necessary libraries as shown in Figure 4

- `matplotlib.pyplot as plt`: Matplotlib is a popular library for creating static, animated, and interactive visualizations in Python.
- `numpy as np`: NumPy is used for numerical operations and provides support for arrays and matrices.
- `pandas as pd`: Pandas is used for data manipulation and analysis, particularly for working with structured data in tabular format.
- `plotly.express as px`: Plotly is a library for creating interactive and visually appealing plots and charts.
- `seaborn as sns`: Seaborn is a high-level interface for creating informative and attractive statistical graphics.
- `matplotlib.dates as mdates`: A module within Matplotlib that provides tools for working with date and time data in plots.
- `deap`: A library for evolutionary algorithms and genetic programming.



```python
import random
import time
from multiprocessing import Pool

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import plotly.express as px
import seaborn as sns
import matplotlib.dates as mdates
from deap import base, creator, tools, algorithms
from gplearn.genetic import SymbolicRegressor
from sklearn.compose import ColumnTransformer
from sklearn.ensemble import RandomForestRegressor
from sklearn.impute import SimpleImputer
from sklearn.linear_model import LinearRegression
from sklearn.manifold import TSNE
from sklearn.metrics import mean_absolute_error, r2_score, median_absolute_error, mean_squared_error
from sklearn.model_selection import cross_val_score, train_test_split
from sklearn.neighbors import KNeighborsRegressor
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import LabelEncoder, StandardScaler, OneHotEncoder
from xgboost import XGBRegressor
from scipy import stats
from wordcloud import WordCloud
```

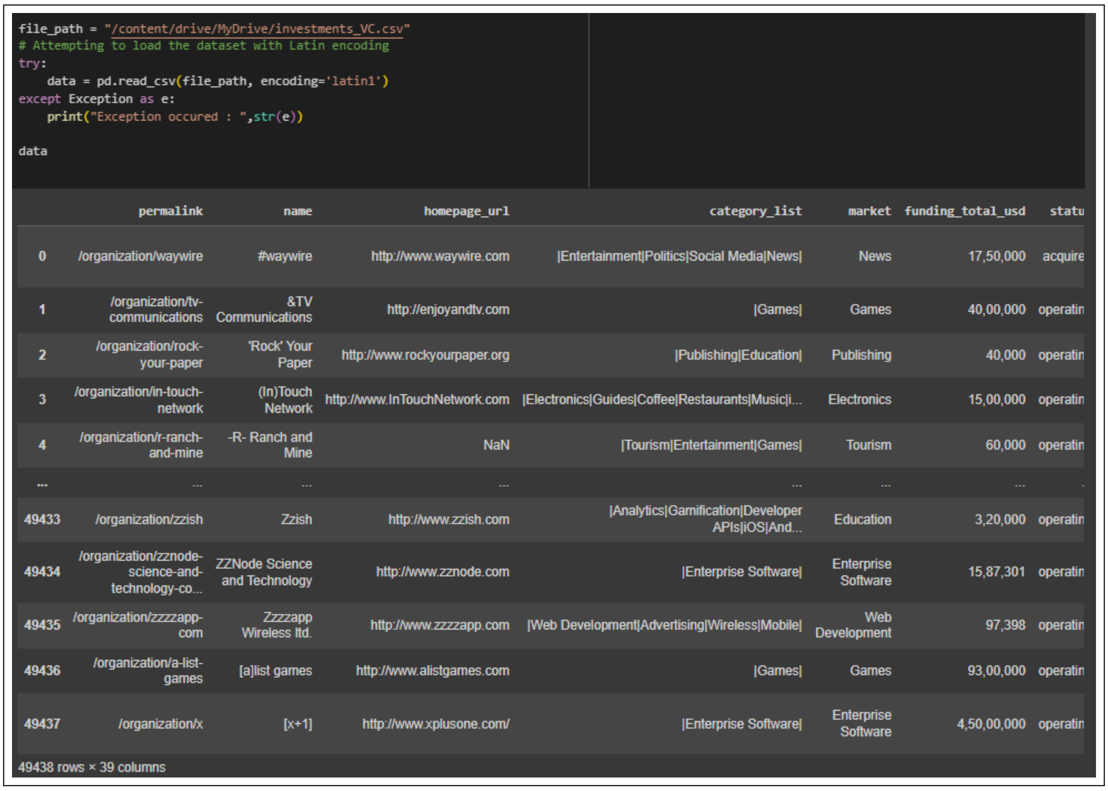
Figure 4: Importing Libraries

- `gplearn.genetic`: A library for symbolic regression using genetic programming.
- `sklearn.compose`: Part of scikit-learn, used for composing transformers and estimators into pipelines.
- `sklearn.ensemble`: Scikit-learn's ensemble methods library, which includes techniques like random forests.
- `sklearn.impute`: Provides tools for imputing missing data in scikit-learn pipelines.
- `sklearn.linear_model`: Scikit-learn's linear modeling tools, including linear regression.
- `sklearn.manifold`: Part of scikit-learn, used for manifold learning techniques like t-SNE.
- `sklearn.metrics`: Scikit-learn's library for evaluating model performance with various metrics.
- `sklearn.model_selection`: Provides tools for model selection, including cross-validation and train-test splitting.
- `sklearn.neighbors`: Scikit-learn's k-neighbors-based methods, including k-nearest neighbors regression.
- `sklearn.pipeline`: Part of scikit-learn, used for creating machine learning pipelines.
- `sklearn.preprocessing`: Scikit-learn's library for data preprocessing, including label encoding, standard scaling, and one-hot encoding.

- `xgboost`: A library for gradient boosting, often used for predictive modeling.
- `scipy.stats`: Part of SciPy, provides statistical functions and distributions.
- `wordcloud`: A library for creating word clouds, often used for visualizing word frequencies in text data.

3.4 Loading the dataset

Loading the dataset and printing the dataset to understand the features present. Figure 5



```
file_path = "/content/drive/MyDrive/investments_VC.csv"
# Attempting to load the dataset with Latin encoding
try:
    data = pd.read_csv(file_path, encoding='latin1')
except Exception as e:
    print("Exception occured : ",str(e))

data
```

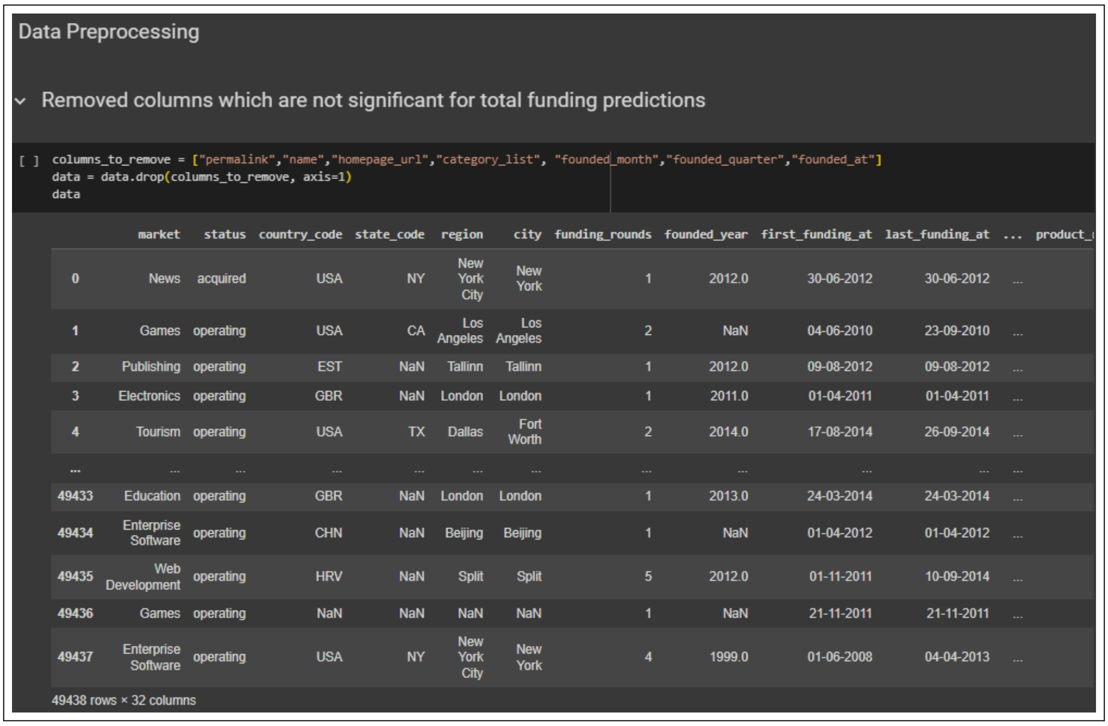
	permalink	name	homepage_url	category_list	market	funding_total_usd	statu
0	/organization/waywire	#waywire	http://www.waywire.com	|Entertainment|Politics|Social Media|News|	News	17,50,000	acquire
1	/organization/tv-communications	&TV Communications	http://enjoyandtv.com	|Games|	Games	40,00,000	operatin
2	/organization/rock-your-paper	'Rock' Your Paper	http://www.rockyourpaper.org	|Publishing|Education|	Publishing	40,000	operatin
3	/organization/in-touch-network	(In)Touch Network	http://www.InTouchNetwork.com	|Electronics|Guides|Coffee|Restaurants|Music|i...	Electronics	15,00,000	operatin
4	/organization/r-ranch-and-mine	-R- Ranch and Mine	NaN	|Tourism|Entertainment|Games|	Tourism	60,000	operatin
...	...	...	...	...	...	...	.
49433	/organization/zzish	Zzish	http://www.zzish.com	|Analytics|Gamification|Developer APIs|iOS|And...	Education	3,20,000	operatin
49434	/organization/zznode-science-and-technology-co...	ZZNode Science and Technology	http://www.zznode.com	|Enterprise Software|	Enterprise Software	15,87,301	operatin
49435	/organization/zzzzapp-com	Zzzzapp Wireless ltd.	http://www.zzzzapp.com	|Web Development|Advertising|Wireless|Mobile|	Web Development	97,398	operatin
49436	/organization/a-list-games	[a]list games	http://www.alistgames.com	|Games|	Games	93,00,000	operatin
49437	/organization/x	[x+1]	http://www.xplusone.com/	|Enterprise Software|	Enterprise Software	4,50,00,000	operatin

49438 rows × 39 columns

Figure 5: Loading Dataset

3.5 Data preprocessing

Removing features that are not important for predicting the total funding USD. Figure 6 shows these features are removed from the dataset as it is not required for total funding prediction "permalink", "name", "homepage_url" and a few features like "category_list, "founded_month", "founded_quarter", "founded_at" are redundant and repetitive in other features present in the dataset. After this many other preprocessing were performed on the dataset and performed in-depth analysis of the dataset through extensive exploratory data analysis.



Data Preprocessing

Removed columns which are not significant for total funding predictions

```
[ ] columns_to_remove = ["permalink","name","homepage_url","category_list", "founded_month","founded_quarter","founded_at"]
    data = data.drop(columns_to_remove, axis=1)
    data
```

	market	status	country_code	state_code	region	city	funding_rounds	founded_year	first_funding_at	last_funding_at	...	product_
0	News	acquired	USA	NY	New York City	New York	1	2012.0	30-06-2012	30-06-2012	...	
1	Games	operating	USA	CA	Los Angeles	Los Angeles	2	NaN	04-06-2010	23-09-2010	...	
2	Publishing	operating	EST	NaN	Tallinn	Tallinn	1	2012.0	09-08-2012	09-08-2012	...	
3	Electronics	operating	GBR	NaN	London	London	1	2011.0	01-04-2011	01-04-2011	...	
4	Tourism	operating	USA	TX	Dallas	Fort Worth	2	2014.0	17-08-2014	26-09-2014	...	
...	...	...	...	...	...	...	...	...	...	...	...	
49433	Education	operating	GBR	NaN	London	London	1	2013.0	24-03-2014	24-03-2014	...	
49434	Enterprise Software	operating	CHN	NaN	Beijing	Beijing	1	NaN	01-04-2012	01-04-2012	...	
49435	Web Development	operating	HRV	NaN	Split	Split	5	2012.0	01-11-2011	10-09-2014	...	
49436	Games	operating	NaN	NaN	NaN	NaN	1	NaN	21-11-2011	21-11-2011	...	
49437	Enterprise Software	operating	USA	NY	New York City	New York	4	1999.0	01-06-2008	04-04-2013	...	

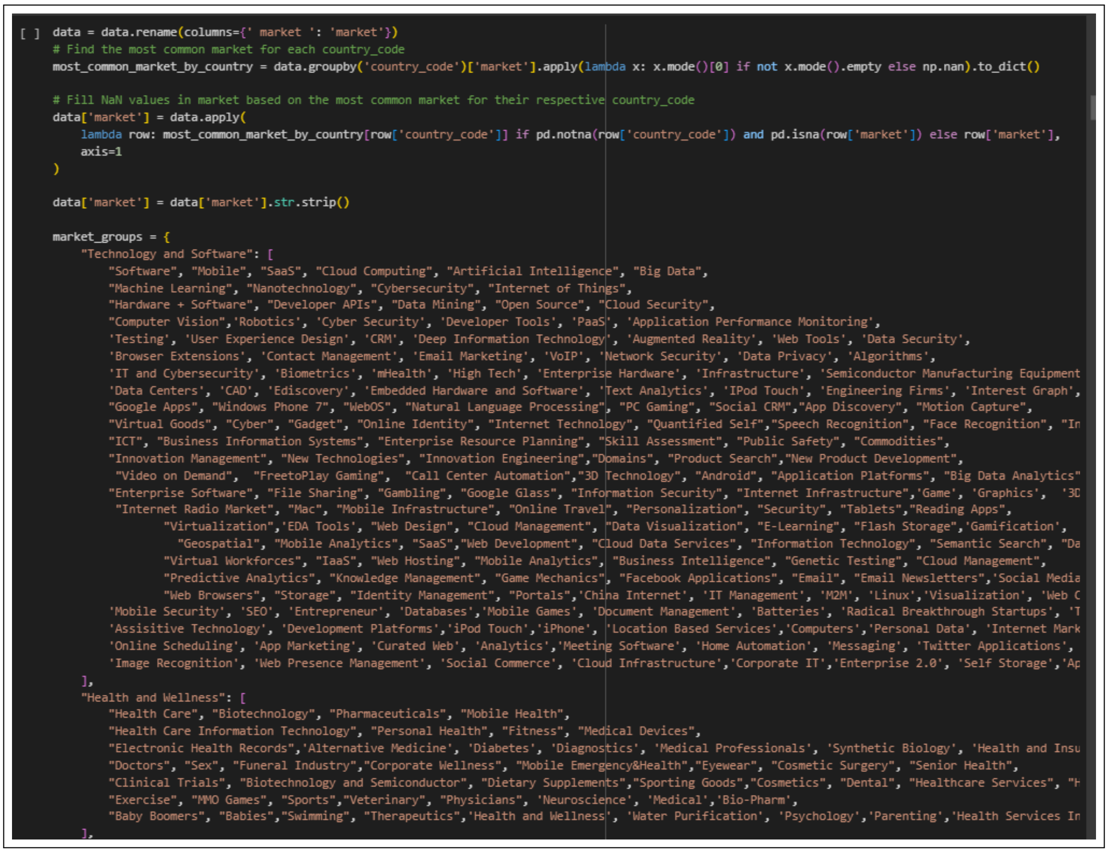
49438 rows × 32 columns

Figure 6: Removing unnecessary columns

3.5.1 Grouping the market into market group bins to understand as sectors

There is a lot of market present in the dataset and it would be easier to analyze and understand different markets when grouped into the market sector each market sector contains a list of markets that is more relevant to that market sector as shown in figure 7. Market Groups are as below:

- Telecommunications
- Travel and Hospitality
- Others
- Technology and Software
- Marketing and Advertising
- Commerce and Retail
- Health and Wellness
- Real Estate and Construction
- Education and Training
- Food and Beverages
- Media and Entertainment
- Transportation and Mobility



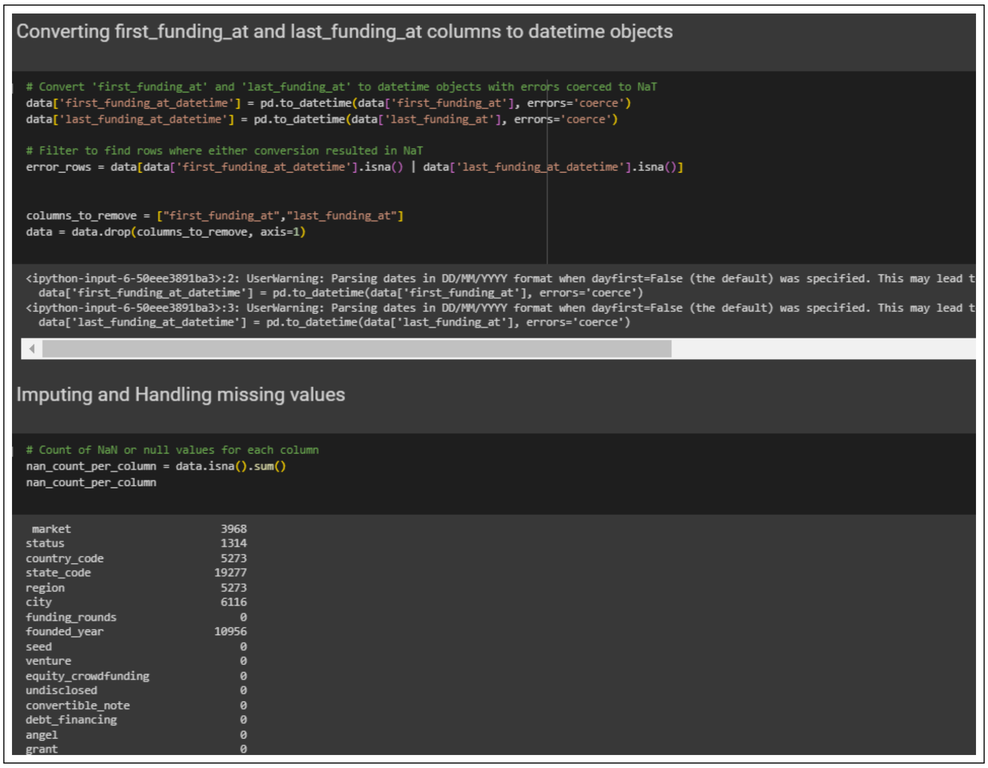
```
[ ]  data = data.rename(columns={' market ': 'market'})
     # Find the most common market for each country_code
     most_common_market_by_country = data.groupby('country_code')['market'].apply(lambda x: x.mode()[0] if not x.mode().empty else np.nan).to_dict()

     # Fill NaN values in market based on the most common market for their respective country_code
     data['market'] = data.apply(
         lambda row: most_common_market_by_country[row['country_code']] if pd.notna(row['country_code']) and pd.isna(row['market']) else row['market'],
         axis=1
     )

     data['market'] = data['market'].str.strip()

     market_groups = {
         "Technology and Software": [
             "Software", "Mobile", "SaaS", "Cloud Computing", "Artificial Intelligence", "Big Data",
             "Machine Learning", "Nanotechnology", "Cybersecurity", "Internet of Things",
             "Hardware + Software", "Developer APIs", "Data Mining", "Open Source", "Cloud Security",
             "Computer Vision",'Robotics', 'Cyber Security', 'Developer Tools', 'PaaS', 'Application Performance Monitoring',
             'Testing', 'User Experience Design', 'CRM', 'Deep Information Technology', 'Augmented Reality', 'Web Tools', 'Data Security',
             'Browser Extensions', 'Contact Management', 'Email Marketing', 'VoIP', 'Network Security', 'Data Privacy', 'Algorithms',
             'IT and Cybersecurity', 'Biometrics', 'mHealth', 'High Tech', 'Enterprise Hardware', 'Infrastructure', 'Semiconductor Manufacturing Equipment
             'Data Centers', 'CAD', 'Ediscovery', 'Embedded Hardware and Software', 'Text Analytics', 'IPod Touch', 'Engineering Firms', 'Interest Graph',
             "Google Apps", "Windows Phone 7", "WebOS", "Natural Language Processing", "PC Gaming", "Social CRM","App Discovery", "Motion Capture",
             "Virtual Goods", "Cyber", "Gadget", "Online Identity", "Internet Technology", "Quantified Self","Speech Recognition", "Face Recognition", "In
             "ICT", "Business Information Systems", "Enterprise Resource Planning", "Skill Assessment", "Public Safety", "Commodities",
             "Innovation Management", "New Technologies", "Innovation Engineering","Domains", "Product Search","New Product Development",
              "Video on Demand",  "FreetoPlay Gaming",  "Call Center Automation","3D Technology", "Android", "Application Platforms", "Big Data Analytics"
             "Enterprise Software", "File Sharing", "Gambling", "Google Glass", "Information Security", "Internet Infrastructure",'Game', 'Graphics',  '3D
              "Internet Radio Market", "Mac", "Mobile Infrastructure", "Online Travel", "Personalization", "Security", "Tablets","Reading Apps",
                     "Virtualization",'EDA Tools', "Web Design", "Cloud Management", "Data Visualization", "E-Learning", "Flash Storage",'Gamification',
                       "Geospatial", "Mobile Analytics", "SaaS","Web Development", "Cloud Data Services", "Information Technology", "Semantic Search", "Da
                     "Virtual Workforces", "IaaS", "Web Hosting", "Mobile Analytics", "Business Intelligence", "Genetic Testing", "Cloud Management",
                     "Predictive Analytics", "Knowledge Management", "Game Mechanics", "Facebook Applications", "Email", "Email Newsletters",'Social Media
                     "Web Browsers", "Storage", "Identity Management", "Portals",'China Internet', 'IT Management', 'M2M', 'Linux','Visualization', 'Web C
             'Mobile Security', 'SEO', 'Entrepreneur', 'Databases','Mobile Games', 'Document Management', 'Batteries', 'Radical Breakthrough Startups', 'T
             'Assisitive Technology', 'Development Platforms','iPod Touch','iPhone', 'Location Based Services','Computers','Personal Data', 'Internet Mark
             'Online Scheduling', 'App Marketing', 'Curated Web', 'Analytics','Meeting Software', 'Home Automation', 'Messaging', 'Twitter Applications',
             'Image Recognition', 'Web Presence Management', 'Social Commerce', 'Cloud Infrastructure','Corporate IT','Enterprise 2.0', 'Self Storage','Ap
         ],
         "Health and Wellness": [
             "Health Care", "Biotechnology", "Pharmaceuticals", "Mobile Health",
             "Health Care Information Technology", "Personal Health", "Fitness", "Medical Devices",
             "Electronic Health Records",'Alternative Medicine', 'Diabetes', 'Diagnostics', 'Medical Professionals', 'Synthetic Biology', 'Health and Insu
             "Doctors", "Sex", "Funeral Industry","Corporate Wellness", "Mobile Emergency&Health","Eyewear", "Cosmetic Surgery", "Senior Health",
             "Clinical Trials", "Biotechnology and Semiconductor", "Dietary Supplements","Sporting Goods","Cosmetics", "Dental", "Healthcare Services", "H
             "Exercise", "MMO Games", "Sports","Veterinary", "Physicians", 'Neuroscience', 'Medical','Bio-Pharm',
             "Baby Boomers", "Babies","Swimming", "Therapeutics",'Health and Wellness', 'Water Purification', 'Psychology','Parenting','Health Services In
         ],
```

Figure 7: Market to Market Group

- Financial Services
- Energy and Environment
- Manufacturing and Industrial
- Other

3.5.2 Handling state city missing data using external mapping

Several state code values were empty but the city value was present. So by using an external mapper that maps the city to its state code values are imputed as shown in figure 8. This sample of code shows several stages in the preprocessing and imputation of data for a dataset. It loads a DataFrame with investment data and a CSV file with state and city information first. Common suffixes like “City” and “County” are removed from the city names to make them more pristine. After that, it updates the investment dataset’s missing state codes and builds a mapping from city to state code using the state-city dataset that was loaded. Furthermore, it uses the ‘first_funding_at_datetime’ column to impute missing values in the ‘founded_year’ column. ‘state_code’ and ‘city’ values that are lacking are further imputed by the code by mapping them to the most prevalent values in individual groupings. Lastly, it reports the count of null or missing values for each column in the dataset and imputes missing “status” values using the mode of the ”status” column. These procedures guarantee that the dataset is full and cleaner for the analysis that comes after.



Converting first_funding_at and last_funding_at columns to datetime objects

```
# Convert 'first_funding_at' and 'last_funding_at' to datetime objects with errors coerced to NaT
data['first_funding_at_datetime'] = pd.to_datetime(data['first_funding_at'], errors='coerce')
data['last_funding_at_datetime'] = pd.to_datetime(data['last_funding_at'], errors='coerce')

# Filter to find rows where either conversion resulted in NaT
error_rows = data[data['first_funding_at_datetime'].isna() | data['last_funding_at_datetime'].isna()]


columns_to_remove = ["first_funding_at","last_funding_at"]
data = data.drop(columns_to_remove, axis=1)
```

```
<ipython-input-6-50eee3891ba3>:2: UserWarning: Parsing dates in DD/MM/YYYY format when dayfirst=False (the default) was specified. This may lead t
  data['first_funding_at_datetime'] = pd.to_datetime(data['first_funding_at'], errors='coerce')
<ipython-input-6-50eee3891ba3>:3: UserWarning: Parsing dates in DD/MM/YYYY format when dayfirst=False (the default) was specified. This may lead t
  data['last_funding_at_datetime'] = pd.to_datetime(data['last_funding_at'], errors='coerce')
```

Imputing and Handling missing values

```
# Count of NaN or null values for each column
nan_count_per_column = data.isna().sum()
nan_count_per_column
```

```
 market                    3968
status                    1314
country_code              5273
state_code               19277
region                    5273
city                      6116
funding_rounds               0
founded_year             10956
seed                         0
venture                      0
equity_crowdfunding          0
undisclosed                  0
convertible_note             0
debt_financing               0
angel                        0
grant                        0
```

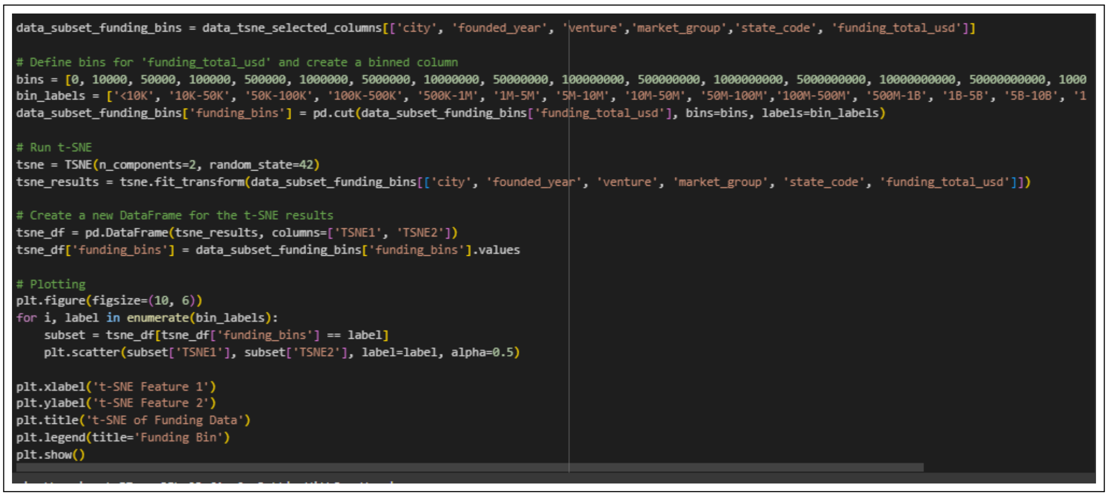
Figure 8: Handling missing data using imputing

3.5.3 TSNE

Code as shown in Figure 9 to generate the TSNE which is used for understanding the dataset for dimensionality reduction and viewing the dataset in 2D graph.

- ‘city’
- ‘founded_year’
- ‘venture’
- ‘market_group’
- ‘state_code’
- ‘funding_total_usd’

Above are among the company data subsets that are first chosen and stored in data subset funding bins. After that, this subset is divided into bins according to the ‘funding_total_usd’ using predetermined ranges; a new column called ‘funding_bins’ is then created with names for these ranges. The high-dimensional data is then reduced to two dimensions using a t-SNE analysis on the subset, which helps to visualize the data’s patterns and clusters. The financing bins that correspond to the two-dimensional characteristics that are obtained using t-SNE are saved in a DataFrame called tsne_df. Ultimately, a scatter plot is made to show the clusters of data points, with each point colored according to its funding bin. This gives an overview of the distribution of funding amounts among the companies and how they relate to each other.



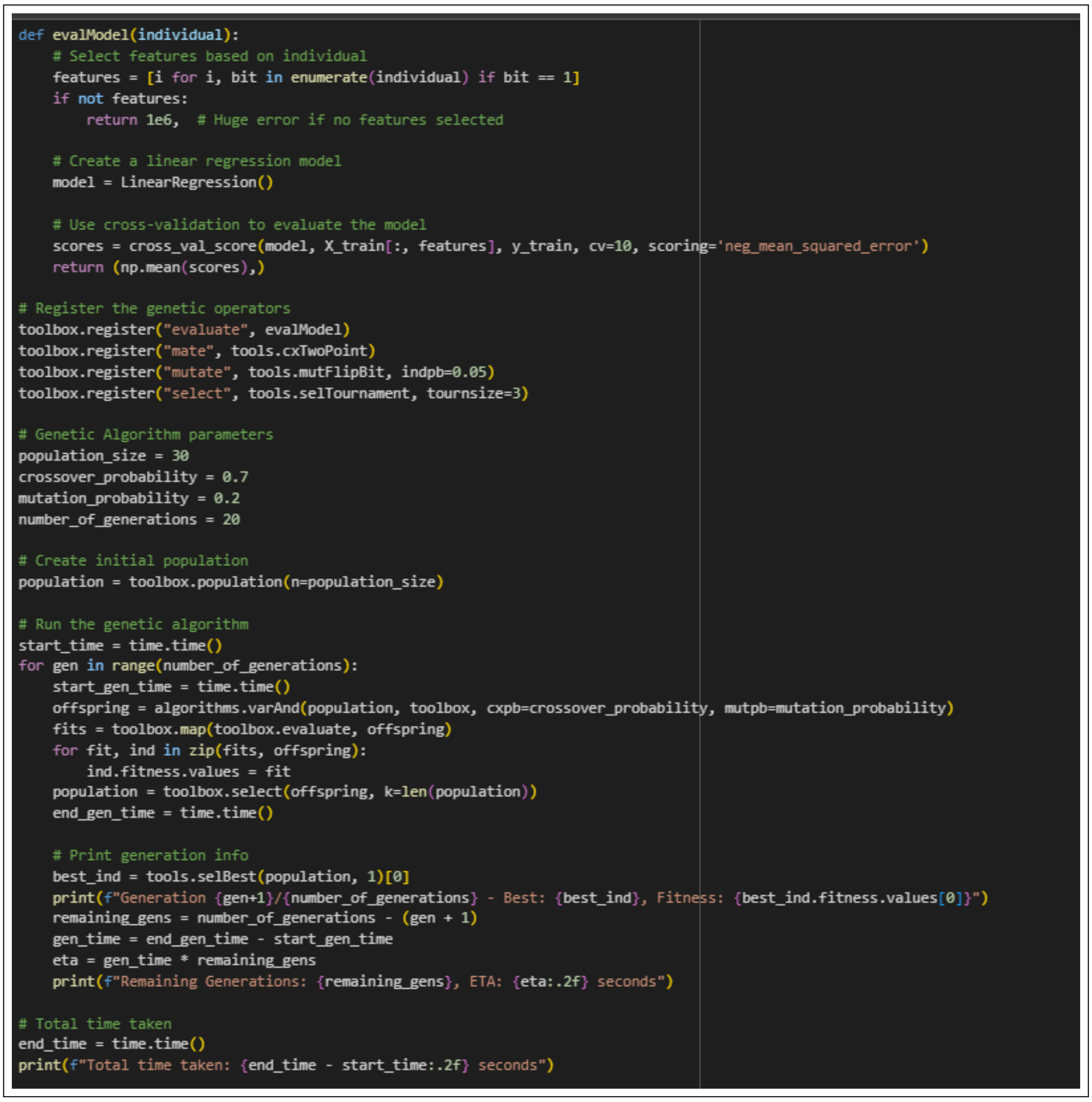
```
data_subset_funding_bins = data_tsne_selected_columns[['city', 'founded_year', 'venture','market_group','state_code', 'funding_total_usd']]

# Define bins for 'funding_total_usd' and create a binned column
bins = [0, 10000, 50000, 100000, 500000, 1000000, 5000000, 10000000, 50000000, 100000000, 500000000, 1000000000, 5000000000, 10000000000, 50000000000, 1000
bin_labels = ['<10K', '10K-50K', '50K-100K', '100K-500K', '500K-1M', '1M-5M', '5M-10M', '10M-50M', '50M-100M','100M-500M', '500M-1B', '1B-5B', '5B-10B', '1
data_subset_funding_bins['funding_bins'] = pd.cut(data_subset_funding_bins['funding_total_usd'], bins=bins, labels=bin_labels)

# Run t-SNE
tsne = TSNE(n_components=2, random_state=42)
tsne_results = tsne.fit_transform(data_subset_funding_bins[['city', 'founded_year', 'venture', 'market_group', 'state_code', 'funding_total_usd']])

# Create a new DataFrame for the t-SNE results
tsne_df = pd.DataFrame(tsne_results, columns=['TSNE1', 'TSNE2'])
tsne_df['funding_bins'] = data_subset_funding_bins['funding_bins'].values

# Plotting
plt.figure(figsize=(10, 6))
for i, label in enumerate(bin_labels):
    subset = tsne_df[tsne_df['funding_bins'] == label]
    plt.scatter(subset['TSNE1'], subset['TSNE2'], label=label, alpha=0.5)

plt.xlabel('t-SNE Feature 1')
plt.ylabel('t-SNE Feature 2')
plt.title('t-SNE of Funding Data')
plt.legend(title='Funding Bin')
plt.show()
```

Figure 9: TSNE

3.5.4 Model building

A code sample of one of the model buildings is given in Figure 10. The provided code describes how to pick features for a linear regression model using a genetic algorithm (GA) (Dasgupta et al.; 2013). Given that the goal is to minimize mistakes, a unique ‘FitnessMin’ class with negative weights has been developed. Furthermore defined is an ‘Individual’ class that represents a solution (a set of features). The toolbox comes pre-configured with ways to build individuals and populations, evaluate fitness using a custom function called evalModel, and produce binary attributes (to determine whether to include each characteristic). Using cross-validation on the training data, this function chooses features that match the ones in each person’s binary string. If no features are selected, a large error is issued.

The population size, crossover probability, mutation probability, and number of generations are established, and the genetic operators for mating, mutating, and selection are defined. The algorithm determines the best individuals based on their fitness and runs for the designated generations, estimating the time to completion. Lastly, it publishes the total time as well as the most attractive and fit person. To reduce the prediction error, this procedure automatically chooses the most important features for the linear regression model.



```python
def evalModel(individual):
    # Select features based on individual
    features = [i for i, bit in enumerate(individual) if bit == 1]
    if not features:
        return 1e6,  # Huge error if no features selected

    # Create a linear regression model
    model = LinearRegression()

    # Use cross-validation to evaluate the model
    scores = cross_val_score(model, X_train[:, features], y_train, cv=10, scoring='neg_mean_squared_error')
    return (np.mean(scores),)

# Register the genetic operators
toolbox.register("evaluate", evalModel)
toolbox.register("mate", tools.cxTwoPoint)
toolbox.register("mutate", tools.mutFlipBit, indpb=0.05)
toolbox.register("select", tools.selTournament, tournsize=3)

# Genetic Algorithm parameters
population_size = 30
crossover_probability = 0.7
mutation_probability = 0.2
number_of_generations = 20

# Create initial population
population = toolbox.population(n=population_size)

# Run the genetic algorithm
start_time = time.time()
for gen in range(number_of_generations):
    start_gen_time = time.time()
    offspring = algorithms.varAnd(population, toolbox, cxpb=crossover_probability, mutpb=mutation_probability)
    fits = toolbox.map(toolbox.evaluate, offspring)
    for fit, ind in zip(fits, offspring):
        ind.fitness.values = fit
    population = toolbox.select(offspring, k=len(population))
    end_gen_time = time.time()

    # Print generation info
    best_ind = tools.selBest(population, 1)[0]
    print(f"Generation {gen+1}/{number_of_generations} - Best: {best_ind}, Fitness: {best_ind.fitness.values[0]}")
    remaining_gens = number_of_generations - (gen + 1)
    gen_time = end_gen_time - start_gen_time
    eta = gen_time * remaining_gens
    print(f"Remaining Generations: {remaining_gens}, ETA: {eta:.2f} seconds")

# Total time taken
end_time = time.time()
print(f"Total time taken: {end_time - start_time:.2f} seconds")
```

Figure 10: Model building

References

Arindam235 (2023). Startup investments (crunchbase), https://www.kaggle.com/datasets/arindam235/startup-investments-crunchbase. Accessed: 25/10/2023.

Dasgupta, K., Mandal, B., Dutta, P., Mandal, J. K. and Dam, S. (2013). A genetic algorithm (ga) based load balancing strategy for cloud computing, *Procedia Technology* **10**: 340–347. First International Conference on Computational Intelligence: Modeling Techniques and Applications (CIMTA) 2013.