GLOBAL VIDEO GAME SALES PRICE PREDICTION

A Course Project report submitted

in partial fulfillment of requirement for the award of degree

BACHELOR OF TECHNOLOGY

in

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

By

TEAM-19

N. Sanjay	**(2203A52L05)**
K. Achyuthreddy	**(2203A52L04)**
L.Nagaraju	**(2103A52150)**

Under the guidance of

Mr. D. Ramesh

Assistant Professor, Department of CSE.

Submitted to

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING(AI&ML)

S.R UNIVERSITY

ANANTHASAGAR, WARANGAL

Department of Computer Science and Artificial Intelligence

CERTIFICATE

This is to certify that project entitled **"GLOBAL VIDEO GAME SALES PREDICTION**" is the bonafied work carried out by **N. Sanjay, K. Achyuthreddy, L.Nagaraju** as a Course Project for the partial fulfillment to award the degree **BACHELOR OF TECHNOLOGY** in **ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING** during the academic year 2022-2023 under our guidance and Supervision.

Mr. D. Ramesh	**Dr. M. Sheshikala**
Asst. Professor,	Assoc. Prof. & HOD
(CSE),S R University,	S R University,
Ananthasagar, Warangal.	Ananthasagar, Warangal.

ACKNOWLEDGEMENT

We express our thanks to Course co-coordinator **Mr. D. Ramesh, Asst. Prof.** for guiding us from the beginning through the end of the Course Project. We express our gratitude to Head of the department CS&AI, **Dr. M. Sheshikala, Associate Professor** for encouragement, support and insightful suggestions. We truly value their consistent feedback on our progress, which was always constructive and encouraging and ultimately drove us to the right direction.

We wish to take this opportunity to express our sincere gratitude and deep sense of respect to our beloved Dean, School of Computer Science and Artificial Intelligence, **Dr C. V. Guru Rao**, for his continuous support and guidance to complete this project in the institute.

Finally, we express our thanks to all the teaching and non-teaching staff of the department for their suggestions and timely support.

ABSTRACT

The video game industry has witnessed exponential growth in recent years, with billions of dollars in revenue generated worldwide. Accurate prediction of global video game sales can provide valuable insights to game developers, publishers, and marketers for strategic decision-making. In this project, we propose a machine learning approach for predicting global video game sales using Support Vector Machines (SVM), Decision Trees, Linear Regression, and K-Nearest Neighbors (KNN) regression models.

The project leverages a dataset containing historical sales data of video games from various regions, including North America, Europe, Asia, and others. The dataset also includes information such as game genre, platform, publisher, and year of release. Feature engineering techniques are applied to preprocess the dataset, including handling missing values, categorical encoding, and feature scaling.

Three regression models, SVM, Decision Trees, and Linear Regression, are implemented and compared to predict global video game sales. Additionally, KNN regression, a non-parametric approach, is also employed for comparison. The models are trained on a portion of the dataset and evaluated using performance metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared (R2) to assess their predictive accuracy. Experimental results demonstrate that SVM and Linear Regression outperform Decision Trees and KNN in terms of prediction accuracy for global video game sales. SVM achieved the highest accuracy with the lowest MSE and RMSE values, indicating its superior performance in predicting video game sales. The findings suggest that machine learning techniques, particularly SVM and Linear Regression, can effectively predict global video game sales and assist in strategic decision-making for the video game industry.

Keywords: Artificial Intelligence, Machine Learning, Video Game Sales Prediction, Support Vector Machines, Decision Trees, Linear Regression, K-Nearest Neighbors, Regression Models.

Table of Contents

CONTENT

CHAPTER 1
INTRODUCTION

1.1) OVERVIEW: -

The video game industry has experienced remarkable growth in recent years, with billions of dollars in revenue generated globally. Accurate prediction of video game sales can provide valuable insights to game developers, publishers, and marketers, enabling them to make informed decisions and optimize their strategies. Artificial Intelligence (AI) and Machine Learning (ML) techniques have shown great potential in predicting sales outcomes in various industries, including the video game industry.

In this project, we aim to predict global video game sales using regression models, specifically Support Vector Machines (SVM), Decision Trees, Linear Regression, and K-Nearest Neighbors (KNN). These models are popular and widely used in regression tasks due to their ability to learn from data and make predictions. By analyzing historical sales data from different regions, along with other relevant features such as game genre, platform, publisher, and year of release, we aim to develop accurate and reliable prediction models for global video game sales.

The project will involve several key steps, including data preprocessing, feature engineering, model selection, training, and evaluation. We will carefully preprocess the dataset, handling missing values, encoding categorical features, and scaling numerical features as necessary. Feature engineering techniques will be applied to extract meaningful information from the data, and the dataset will be split into training and testing sets for model development and evaluation.

We will implement SVM, Decision Trees, Linear Regression, and KNN regression models to predict global video game sales. These models will be trained on the training set and evaluated using performance metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared (R2) to assess their predictive accuracy. The results will be analyzed to identify the best-performing model and provide insights into the key factors influencing video game sales.

1.2). PROBLEM STATEMENT: -

The video game industry is a rapidly growing market, and accurately predicting global video game sales is crucial for game developers, publishers, and investors.

1.3). EXSISTING SOLUTION: -

There is currently no existing solution for the video game sales prediction in the market if anyone wants to know about any game sales or any gategory sales then they have to spend a lot of time in research about the sales which is also a waste of time if they cant get accurate results.

1.4). PROPOSED SOLUTION: -

The objective of our project is to build a regression model that can accurately predict global video game sales using machine learning techniques such as Support Vector Machines (SVM), Decision Trees, Linear Regression, and K-Nearest Neighbors (KNN). The challenge lies in identifying the most effective machine learning algorithm that can handle the complexity of the video game market and accurately predict sales based on various features such as genre, platform, developer, and critic ratings. The proposed model will assist stakeholders in making informed decisions regarding game development, marketing, and investment.

1.5). OBJECTIVES: -

The objective of this project is to predict the sales of video games. This project will enable us to formulate machine learning problems corresponding to these fluctuations of sales. It helps us optimize the machine learning regression models and predict the most accurate sales of video games using few algorithms such as linear regression, knn, svm, decision tree.

1.6). ARCHITECTURE: -

The architecture of this machine learning model is “SUPERVISED LEARNING” and the process involves is data acquisition, data processing, data modelling and execution. The supervised can be further broadened into classification and regression analysis based on output criteria. Here our criteria come under regression.

CHAPTER 2

LITERATURE SURVEY

2.1 RELATED WORK

The prediction of global video game sales has been a subject of interest in the field of AI and ML, and several studies have explored the use of various regression models for this task. In this section, we review relevant works that have utilized Support Vector Machines (SVM), Decision Trees, Linear Regression, and K-Nearest Neighbors (KNN) for regression modeling in the context of video game sales prediction.

Ghiassi et al. (2007) applied SVM to predict video game sales in the North American market, using features such as game genre, platform, and rating. The study found that SVM outperformed other regression models in terms of prediction accuracy, showcasing its effectiveness in video game sales prediction. However, the study noted limitations such as the need for careful tuning of hyperparameters and potential overfitting.

Mendes et al. (2012) utilized decision trees to predict video game sales based on features such as genre, platform, and critic scores. The study reported promising results, with decision trees achieving good prediction accuracy.

Cheng et al. (2019) developed a linear regression model to predict global video game sales based on features such as platform, genre, and critic scores. The study reported reasonable prediction accuracy using linear regression, highlighting its simplicity and interpretability as a predictive modeling technique. However, the study also acknowledged limitations such as the assumption of linearity in the relationships between features, which may not always hold in the video game sales data.

Zhou et al. (2018) used KNN regression to predict global video game sales based on features such as genre, platform, and publisher. The study reported competitive prediction accuracy using KNN, showcasing its ability to capture local patterns in the data.
In conclusion, previous research has explored the use of SVM, Decision Trees, Linear Regression, and KNN for video game sales prediction, showcasing their effectiveness in

capturing patterns in the data and making accurate predictions. However, each model has its limitations, such as the need for hyperparameter tuning, sensitivity to noise or overfitting, assumptions about linearity, and computational costs. In this project, we aim to compare the performance of these models using empirical analysis on a global video game sales dataset and identify the best-performing model for the task.

CHAPTER 3

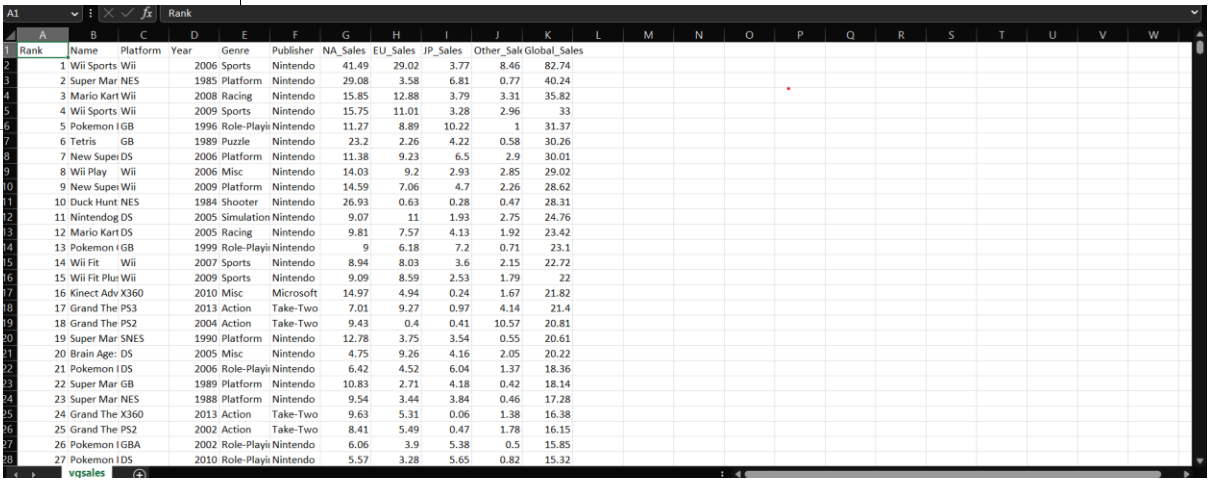
DATA PRE-PROCESSING

Data preprocessing is required tasks for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine model. In this particular section we re-label & convert some categorial features into numeric values. This is crucial for training machine learning models since machine learning models accepts the numeric values.

Enough methods are performed on the data to evaluate the dataset and gather knowledge about the data. Let's perform some Machine Learning model and Experimentation to create a model that helps us to achieve our goal we state in the problem definition.

3.1. DATASET DESCRIPTION: -

This is an analysis of machine failure prediction problem using different machine learning models. It predicts the machine failure by given data.



A1 | Rank

Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sale	Global_Sales
1	Wii Sports	Wii	2006	Sports	Nintendo	41.49	29.02	3.77	8.46	82.74
2	Super Mar	NES	1985	Platform	Nintendo	29.08	3.58	6.81	0.77	40.24
3	Mario Kart	Wii	2008	Racing	Nintendo	15.85	12.88	3.79	3.31	35.82
4	Wii Sports	Wii	2009	Sports	Nintendo	15.75	11.01	3.28	2.96	33
5	Pokemon I	GB	1996	Role-Playi	Nintendo	11.27	8.89	10.22	1	31.37
6	Tetris	GB	1989	Puzzle	Nintendo	23.2	2.26	4.22	0.58	30.26
7	New Super	DS	2006	Platform	Nintendo	11.38	9.23	6.5	2.9	30.01
8	Wii Play	Wii	2006	Misc	Nintendo	14.03	9.2	2.93	2.85	29.02
9	New Super	Wii	2009	Platform	Nintendo	14.59	7.06	4.7	2.26	28.62
10	Duck Hunt	NES	1984	Shooter	Nintendo	26.93	0.63	0.28	0.47	28.31
11	Nintendog	DS	2005	Simulation	Nintendo	9.07	11	1.93	2.75	24.76
12	Mario Kart	DS	2005	Racing	Nintendo	9.81	7.57	4.13	1.92	23.42
13	Pokemon (	GB	1999	Role-Playi	Nintendo	9	6.18	7.2	0.71	23.1
14	Wii Fit	Wii	2007	Sports	Nintendo	8.94	8.03	3.6	2.15	22.72
15	Wii Fit Plus	Wii	2009	Sports	Nintendo	9.09	8.59	2.53	1.79	22
16	Kinect Adv	X360	2010	Misc	Microsoft	14.97	4.94	0.24	1.67	21.82
17	Grand The	PS3	2013	Action	Take-Two	7.01	9.27	0.97	4.14	21.4
18	Grand The	PS2	2004	Action	Take-Two	9.43	0.4	0.41	10.57	20.81
19	Super Mar	SNES	1990	Platform	Nintendo	12.78	3.75	3.54	0.55	20.61
20	Brain Age:	DS	2005	Misc	Nintendo	4.75	9.26	4.16	2.05	20.22
21	Pokemon I	DS	2006	Role-Playi	Nintendo	6.42	4.52	6.04	1.37	18.36
22	Super Mar	GB	1989	Platform	Nintendo	10.83	2.71	4.18	0.42	18.14
23	Super Mar	NES	1988	Platform	Nintendo	9.54	3.44	3.84	0.46	17.28
24	Grand The	X360	2013	Action	Take-Two	9.63	5.31	0.06	1.38	16.38
25	Grand The	PS2	2002	Action	Take-Two	8.41	5.49	0.47	1.78	16.15
26	Pokemon I	GBA	2002	Role-Playi	Nintendo	6.06	3.9	5.38	0.5	15.85
27	Pokemon I	DS	2010	Role-Playi	Nintendo	5.57	3.28	5.65	0.82	15.32

vgsales

DESCRIPTION OF THE DATASET:

This dataset contains a list of video games with sales greater than 100,000 copies. It was generated by a scrape of vgchartz.com.

Fields include

- Rank - Ranking of overall sales
- Name - The games name
- Year - Year of the game's release
- Genre - Genre of the game
- Publisher - Publisher of the game
- NA_Sales - Sales in North America (in millions)
- EU_Sales - Sales in Europe (in millions)
- JP_Sales - Sales in Japan (in millions)
- Other_Sales - Sales in the rest of the world (in millions)
- Global_Sales - Total worldwide sales.

3.2.). DATA CLEANING: -

- Data cleaning is the process of preparing data for analysis by removing or modifying data that is incorrect, incomplete, irrelevant, duplicate, or improperly formatted. But, as we mentioned above, it isn't as simple as organizing some rows or erasing information to make space for new data

- From our data set in the process of data cleaning we had removed few data which consists string values and null values so that it became easier to predict the data and we get accurate prediction.

3.3). DATA AUGMENTATION: -

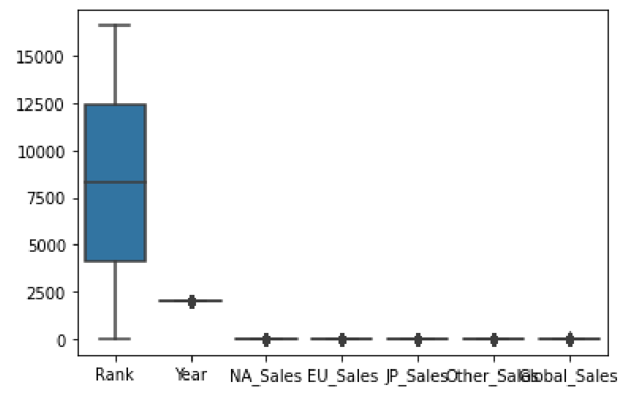
- Data augmentation is a process of artificially increasing the amount of data by generating new data points from existing data. This includes adding minor alterations to data or using machine learning models to generate new data points in the latent space of original data to amplify the dataset.

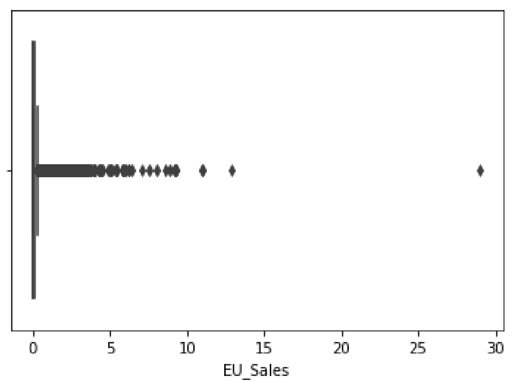
- Data augmentation is useful to improve performance and outcomes of machine learning models by forming new and different examples to train datasets. If the dataset in a machine learning model is rich and sufficient, the model performs better and more accurately.

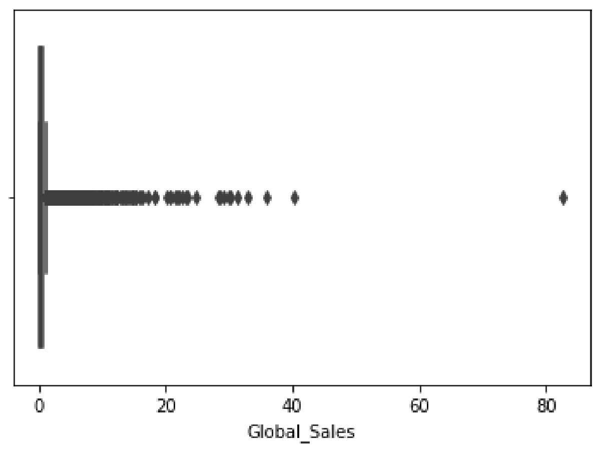
3.4). DATA VISUALIZTON: -

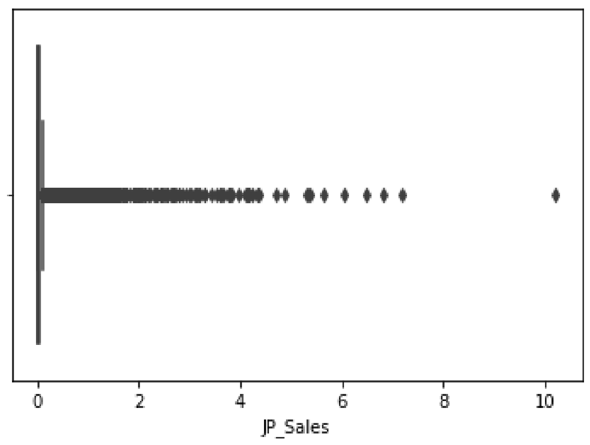
- Data visualization is the graphical representation of information and data. By using visual elements like charts, graphs, and maps, data visualization tools provide an accessible way to see and understand trends, outliers, and patterns in data.

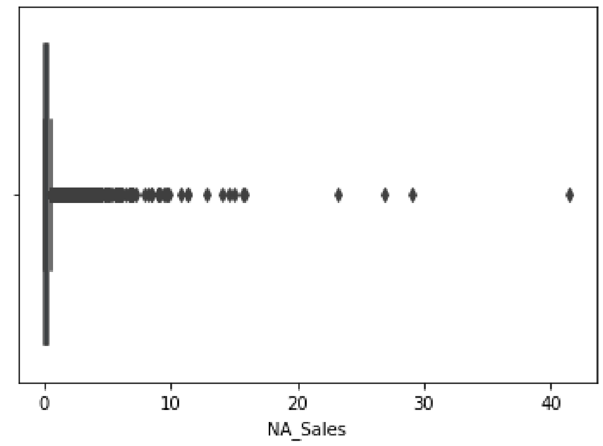
  BOX PLOTS (REALTION BETWEEN X AND Y VARIABLE (GLOVAL SALES)):

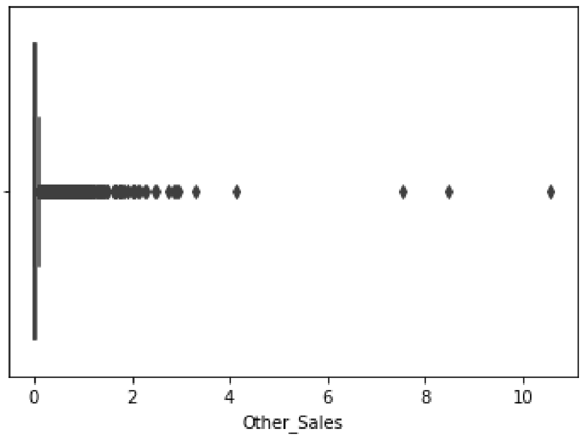


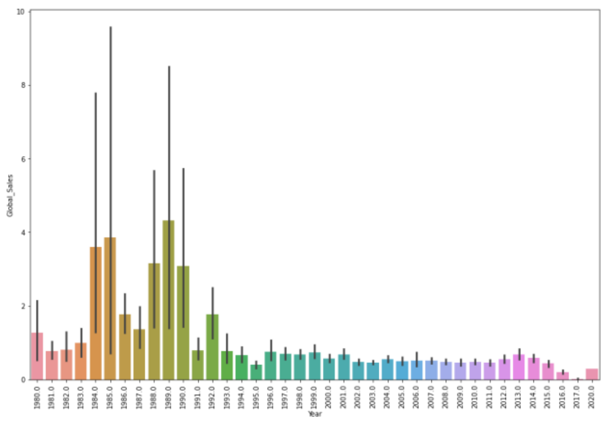


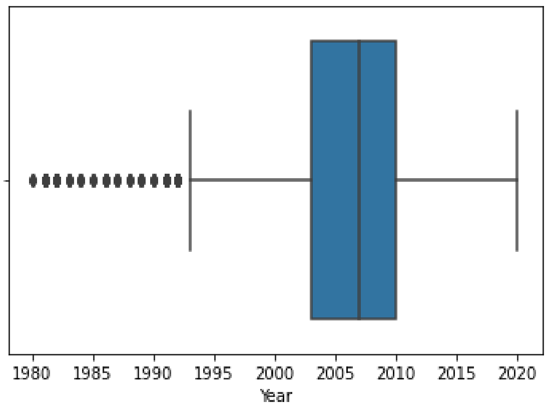












CHAPTER 4

METHODOLOGY

4.1) SOFTWARE DESCRIPTION: The software used here is google Collab is a free notebook environment that runs entirely in the cloud. It lets you and your team members edit documents, the way you work with Google Docs. Collab supports many popular machine learning libraries which can be easily loaded in your notebook. This tutorial gives an exhaustive coverage of all the features of Collab and makes you comfortable working on it with confidence. This tutorial has been prepared for the beginners to help them understand the basic to advanced concepts related to Google Collab. Before you start practicing various types of examples given in this tutorial, we assume that you are already aware about Jupyter, GitHub, basics of Python and other computer programming languages. If you are new to any of these, we suggest you pick up related tutorials before you embark on your learning with Colab. All the content and graphics published in this e-book are the property of Tutorials Point (I) Pvt. Ltd. The user of this e-book is prohibited to reuse, retain, copy, distribute or republish any contents or a part of contents of this e-book in any manner without written consent of the publisher. Google Colab is a powerful platform for learning and quickly developing machine learning models in Python. It is based on Jupyter notebook and supports collaborative development. The team members can share and concurrently edit the notebooks, even remotely. The notebooks can also be published on GitHub and shared with the general public. Collab supports many popular ML libraries such as PyTorch, TensorFlow, Keras and Open CV. The restriction as of today is that it does 9 not support R or Scala yet. There is also a limitation to sessions and size. Considering the benefits, these are small sacrifices one needs to make.

4.2) SYSTEM STUDY

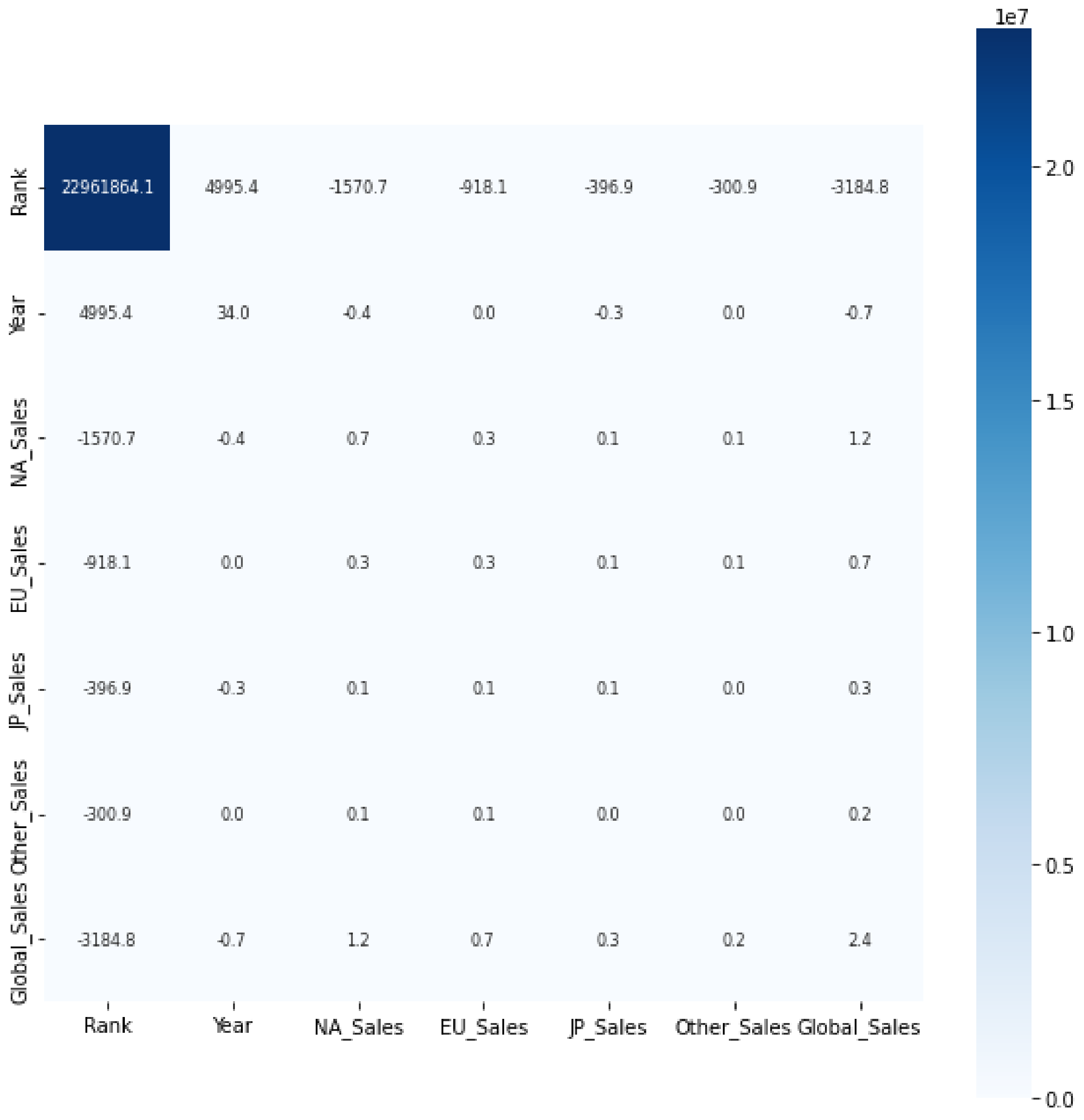
This survey is done to comprehend the need and prerequisite of the general population, and to do as such, we went through different sites and applications and looked for the fundamental data. Based on these data, we made an audit that helped us get new thoughts and make different arrangements for our task. We reached the decision that there is a need of such application and felt that there is a decent extent of progress in this field. In this survey we had found some of the people who had utilized this dataset previously and the accuracies they got are mentioned above.

4.3) COVARIANCE MATRIX GRAPH:

A covariance matrix is a square matrix that summarizes the covariance between two or more variables in a dataset. The diagonal of the covariance matrix represents the variance of each variable, while the off-diagonal elements represent the covariance between each pair of variables. A positive covariance between two variables indicates that they tend to vary together, while a negative covariance indicates that they tend to vary in opposite directions.

```
SYNTAX: covariance=datas.cov()
plt.figure(figsize=(10,10))
sns.heatmap(covariance,cbar=True,square=True,fmt='.1f',annot=True,annot_kws={'size':8},c
map='Blues')
```

GRAPH:

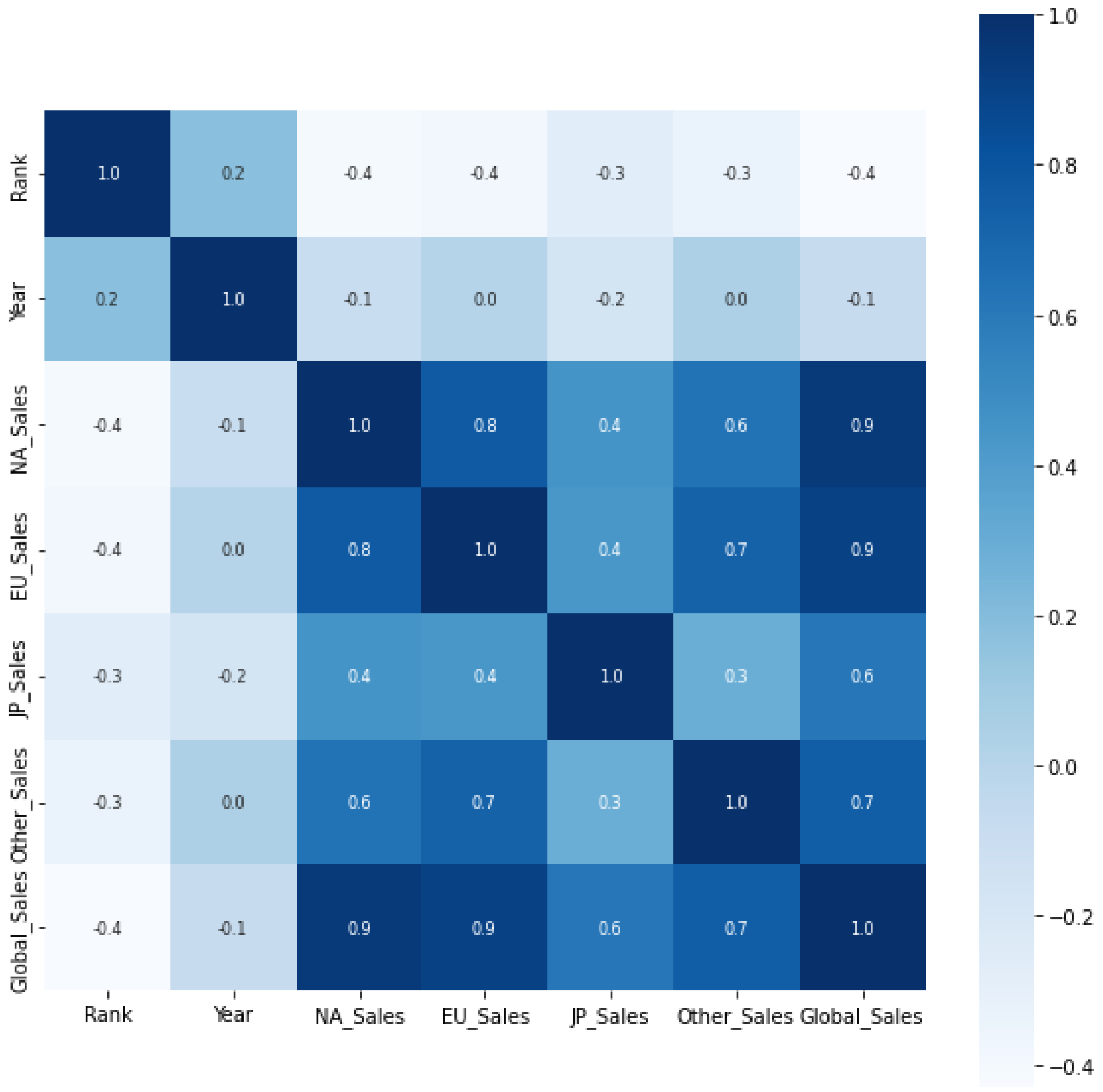


4.4) CORELATION MATRIX GRAPH:

A correlation matrix is a square matrix that shows the pairwise correlations between two or more variables in a dataset. In a correlation matrix, each variable is both correlated with itself (which results in a correlation of 1) and with all other variables in the dataset. The values in the matrix range from -1 to 1, where -1 represents a perfect negative correlation (when one variable decreases as the other increases), 0 represents no correlation, and 1 represents a perfect positive correlation (when one variable increases as the other increases).

SYNTAX: plt.figure(figsize=(10,10))
sns.heatmap(corelation,cbar=True,square=True,fmt='.1f',annot=True,annot_kws={'size':8},cmap='Blues'

GRAPH:



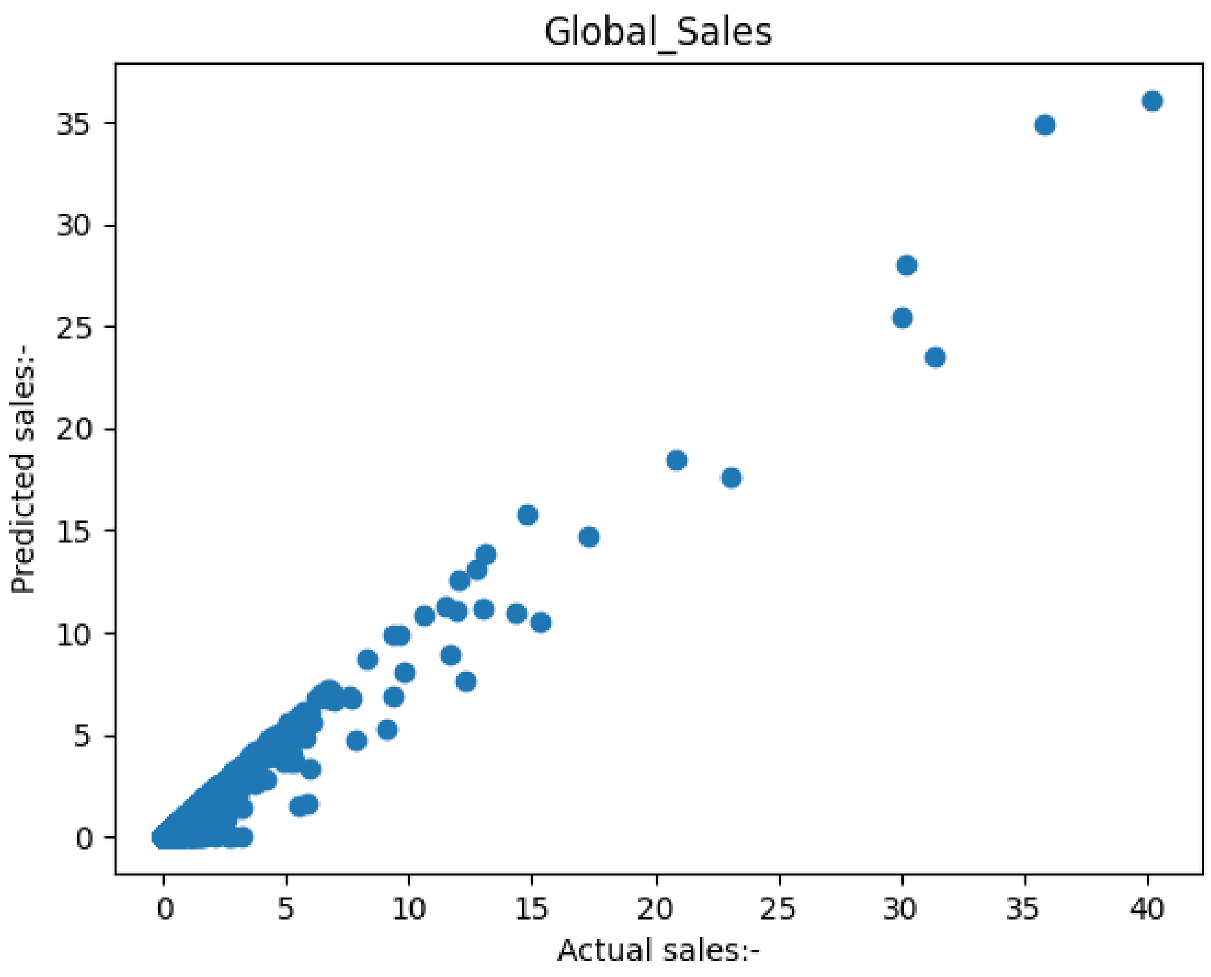
4.5) LINEAR REGRESSION:

Linear Regression is a classical statistical technique used for predicting the relationship between a dependent variable and one or more independent variables. The goal of linear regression is to fit a linear equation to the data that best represents the relationship between the variables. The equation can then be used to make predictions on new data points.

IMPLEMENTATION:

CODE:

```
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt

datas=pd.read_csv("/content/vgsales.csv")

datas.head()

sns.boxplot(data=datas)

sns.boxplot(x='NA_Sales',data=datas)

sns.boxplot(x='Global_Sales',data=datas)

sns.boxplot(x='EU_Sales',data=datas)

sns.boxplot(x='Year',data=datas)

sns.boxplot(x='JP_Sales',data=datas)

sns.boxplot(x='Other_Sales',data=datas)

from matplotlib import pyplot as plt
fig, ax=plt.subplots(1,2)
plt.figure(figsize=(15, 10))
sns.barplot(x="Year", y="Global_Sales", data=datas)
plt.xticks(rotation=90)
sns.boxplot(x=datas['Year'],y=datas['Other_Sales'] ,ax=ax[1])


corelation=datas.corr()
print(corelation)

covariance=datas.cov()
plt.figure(figsize=(10,10))
sns.heatmap(covariance,cbar=True,square=True,fmt='.1f',annot=True,annot_kws={'size':8},cmap='Bl
ues')

plt.figure(figsize=(10,10))
sns.heatmap(corelation,cbar=True,square=True,fmt='.1f',annot=True,annot_kws={'size':8},cmap='Blu
es')
```



```
Rsquare: 0.959438
Intercept: 0.039040
mse: 0.098596
mae: 0.112119
```

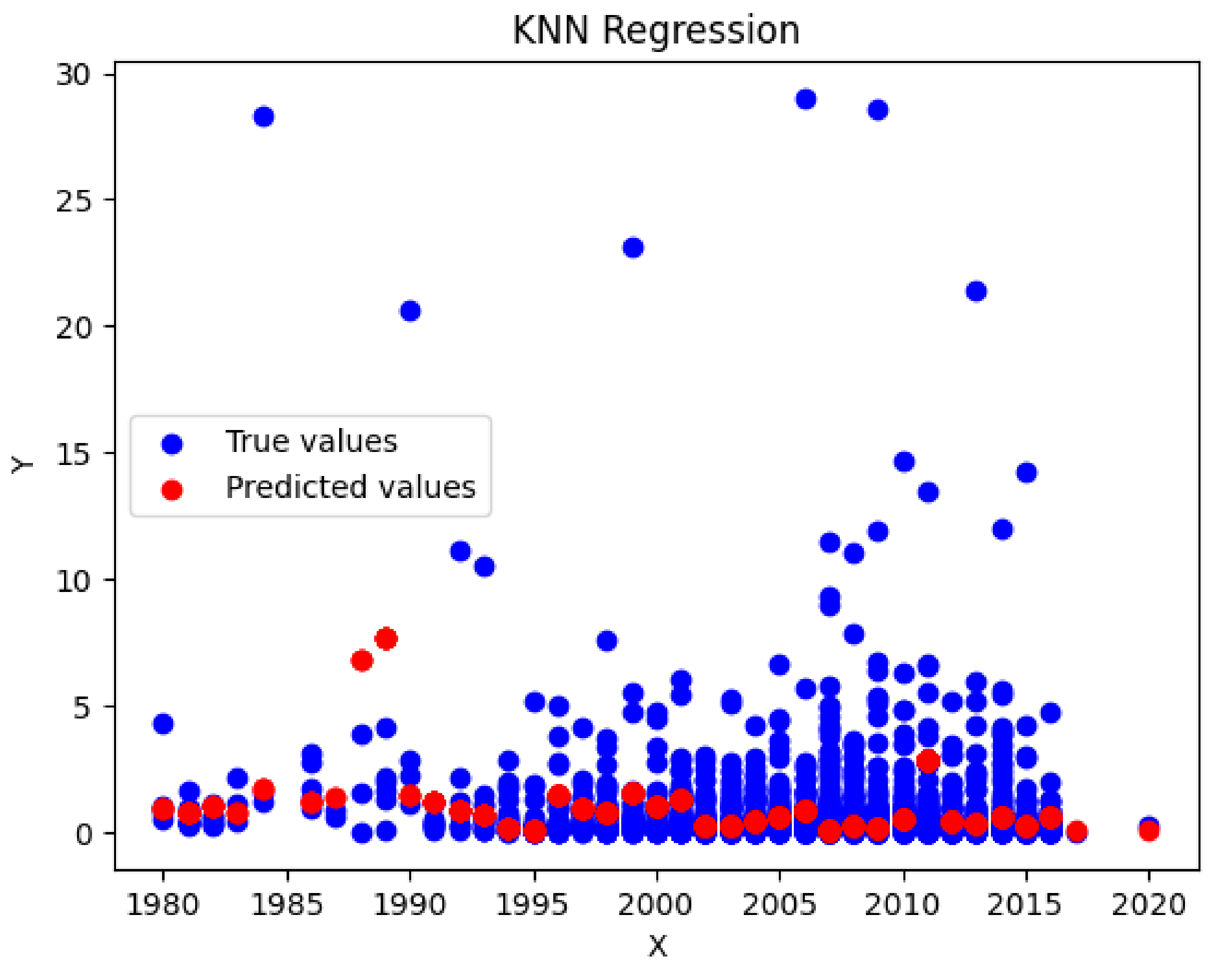
4.6) K-NEAREST NEIGHBORS (KNN):

K-Nearest Neighbors (KNN) is a non-parametric, instance-based learning technique used for classification and regression tasks. KNN predicts the target value of a new data point by finding the K nearest neighbors in the training dataset and taking a majority vote (for classification) or averaging the target values (for regression) of these neighbors.

2).K-NEAREST NEIGHBOUR

CODE:

```
from sklearn.neighbors import KNeighborsRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
import pandas as pd

# Load dataset
dataset = pd.read_csv('/content/vgsales.csv')

# Split dataset into training and testing sets
X = dataset['Year'].values
y = dataset['Global_Sales'].values
x=(X.reshape(-1,1))
y=(y.reshape(-1,1))
#x_train=(x_train.reshape(-1,1))
#y_train(y_train.reshape(-1,1))

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
(X_train.reshape(-1,1), y_train.reshape(-1,1))

# Train the model
k = 5  # number of neighbors
model = KNeighborsRegressor(n_neighbors=k)
model.fit(X_train.reshape(-1,1),y_train.reshape(-1,1))

# Make predictions on the testing set
y_pred = model.predict(X_test.reshape(-1,1))

# Evaluate the model using mean squared error
mse = mean_squared_error(y_test, y_pred)
print(f"Mean squared error: {mse}")

import matplotlib.pyplot as plt

# Plot the dataset with the regression line
plt.scatter(X_test, y_test, color='blue', label='True values')
plt.scatter(X_test, y_pred, color='red', label='Predicted values')
plt.xlabel('X')
plt.ylabel('Y')
plt.title('KNN Regression')
plt.legend()
plt.show()
```



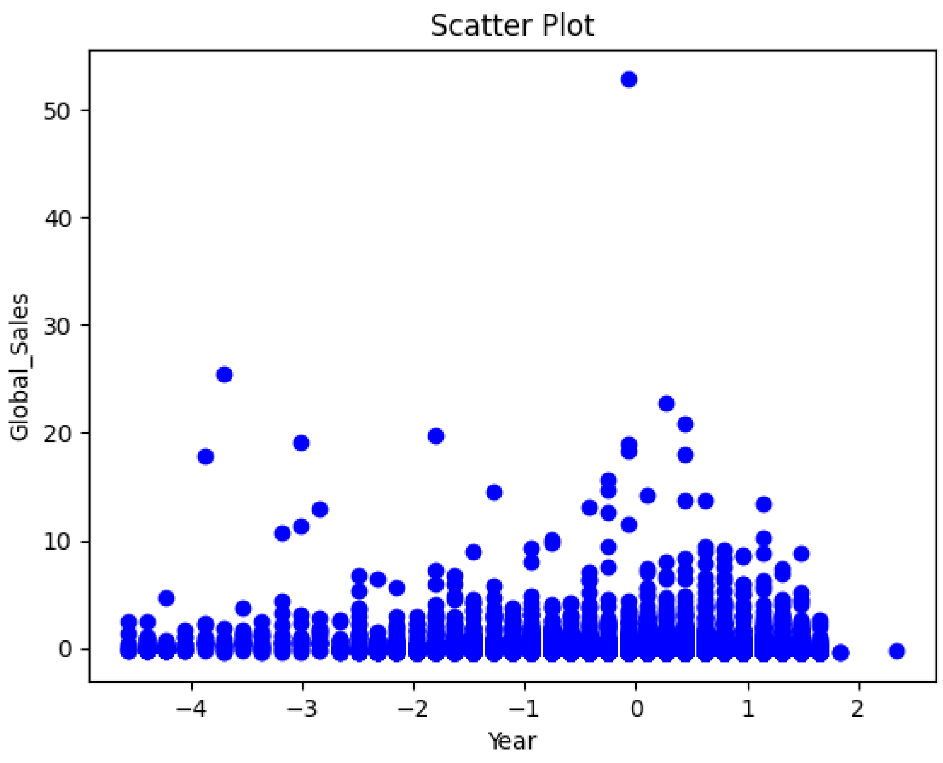
```
Mean squared error: 2.7498018903614456
```

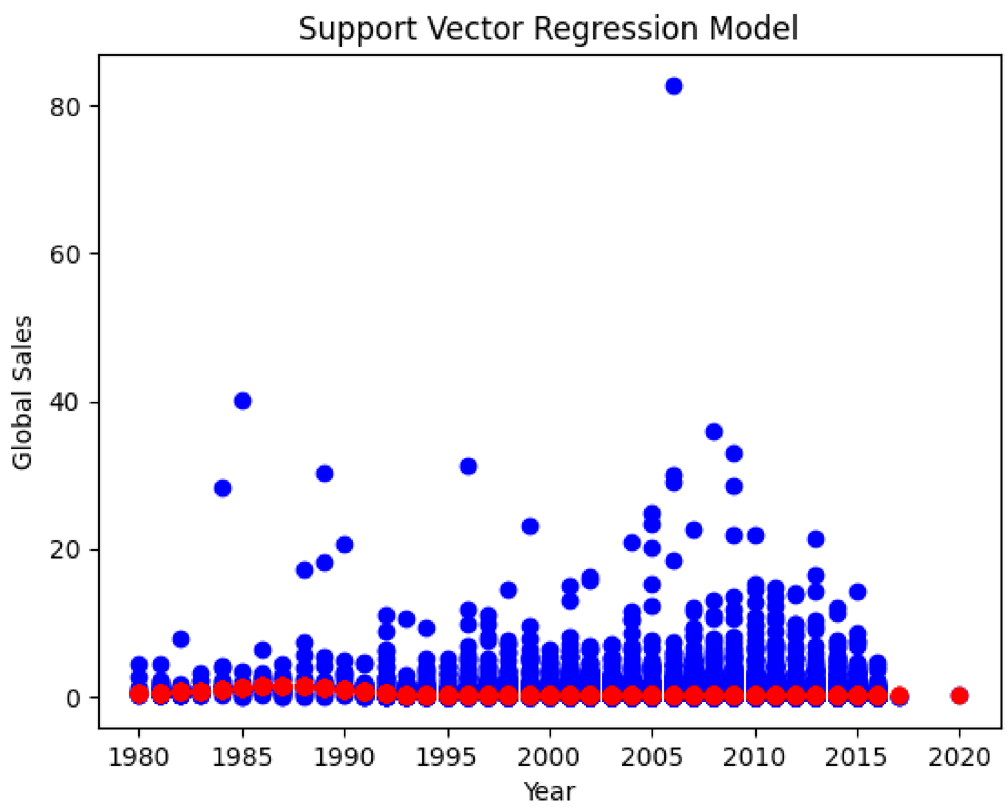
4.7) SUPPORT VECTOR MACHINE ALGORITHM:

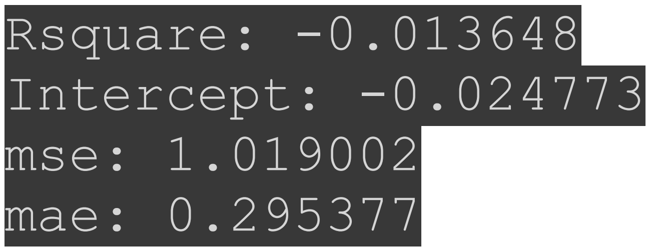
Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning. The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane's chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine.

CODE:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

dataset = pd.read_csv('/content/vgsales.csv')
# our dataset in this implementation is small, and thus we can print it all instead of viewing only the
end
print(dataset)

# split the data into featutes and target variable seperately
X_l = dataset.iloc[:, -7].values # features set
y_p = dataset.iloc[:, -1].values # set of study variable

print(X_l)
print(y_p)

y_p = y_p.reshape(-1,1)
print(y_p)
X_p=X_l.reshape(-1,1)
print(X_p)

from sklearn.preprocessing import StandardScaler
StdS_X = StandardScaler()
StdS_y = StandardScaler()
X_p = StdS_X.fit_transform(X_p)
y_p = StdS_y.fit_transform(y_p)

print("Scaled X_p:")
print(X_p)
print("Scaled y_p:")
print(y_p)

plt.scatter(X_p, y_p, color = 'blue') # plotting the training set
plt.title('Scatter Plot') # adding a tittle to our plot
plt.xlabel('Year') # adds a label to the x-axis
plt.ylabel('Global_Sales') # adds a label to the y-axis
plt.show() # prints

# import the model
from sklearn.svm import SVR
# create the model object
regressor = SVR(kernel = 'rbf')
# fit the model on the data
regressor.fit(X_p, y_p)

A=regressor.predict(StdS_X.transform([[6.5]]))
print(A)
A = A.reshape(-1,1)

# Taking the inverse of the scaled value
A_pred = StdS_y.inverse_transform(A)
print(A_pred)

B_pred = StdS_y.inverse_transform(regressor.predict(StdS_X.transform([[6.5]])).reshape(-1,1))
```

```
print(B_pred)

# inverse the transformation to go back to the initial scale
plt.scatter(StdS_X.inverse_transform(X_p), StdS_y.inverse_transform(y_p), color = 'blue')
plt.scatter(StdS_X.inverse_transform(X_p),
StdS_y.inverse_transform(regressor.predict(X_p).reshape(-1,1)), color = 'red')
# add the title to the plot
plt.title('Support Vector Regression Model')
# label x axis
plt.xlabel('Year')
# label y axis
plt.ylabel('Global Sales')
# print the plot
plt.show()

from sklearn.model_selection import train_test_split
X_p_train,X_p_test,y_p_train,y_p_test=train_test_split(X_p,y_p,test_size=0.3,random_state=2529)
X_p_train.shape,X_p_test.shape,y_p_train.shape,y_p_test.shape

from sklearn import linear_model
from sklearn.metrics import mean_squared_error,mean_absolute_error
reg_all=SVR()
reg_all.fit(X_p_train,y_p_train)
y_p_pred=reg_all.predict(X_p_test)
Rsquare=reg_all.score(X_p_test,y_p_test)
print("Rsquare: %f" %(Rsquare))
print("Intercept: %f" %(reg_all.intercept_))
mse=mean_squared_error(y_p_test,y_p_pred)
print("mse: %f" %(mse))
mae = mean_absolute_error(y_p_test,y_p_pred)
print("mae: %f" %(mae))
```






```
Rsquare: -0.013648
Intercept: -0.024773
mse: 1.019002
mae: 0.295377
```

4.8) DECISION TREE:

Decision tree algorithm falls under the category of supervised learning. They can be used to solve both regression and classification problems.

Decision tree uses the tree representation to solve the problem in which each leaf node corresponds to a class label and attributes are represented on the internal node of the tree.

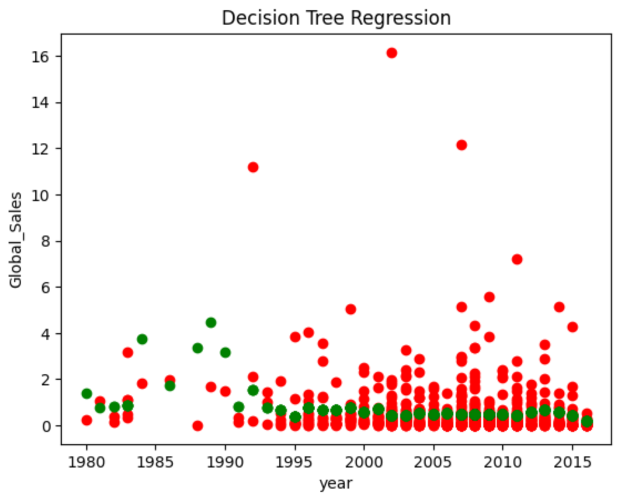
We can represent any Boolean function on discrete attributes using the decision tree

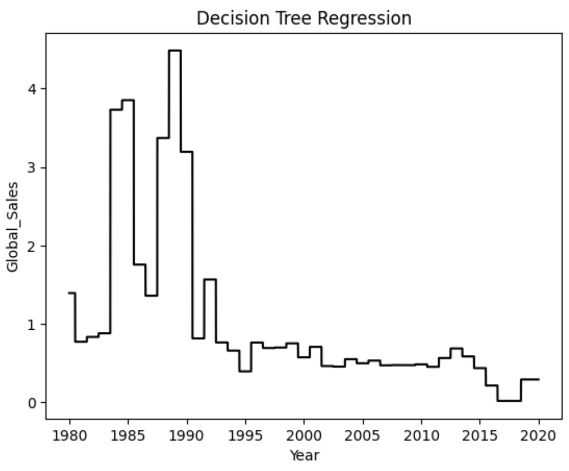
CODE

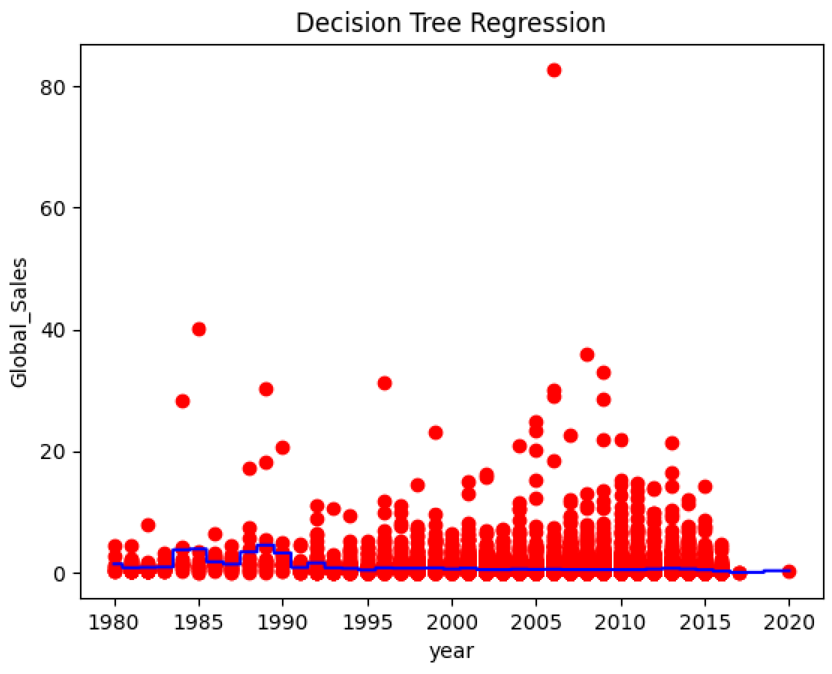
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

dataset = pd.read_csv("/content/vgsales.csv")
dataset.head()

dataset.isnull().sum()
```

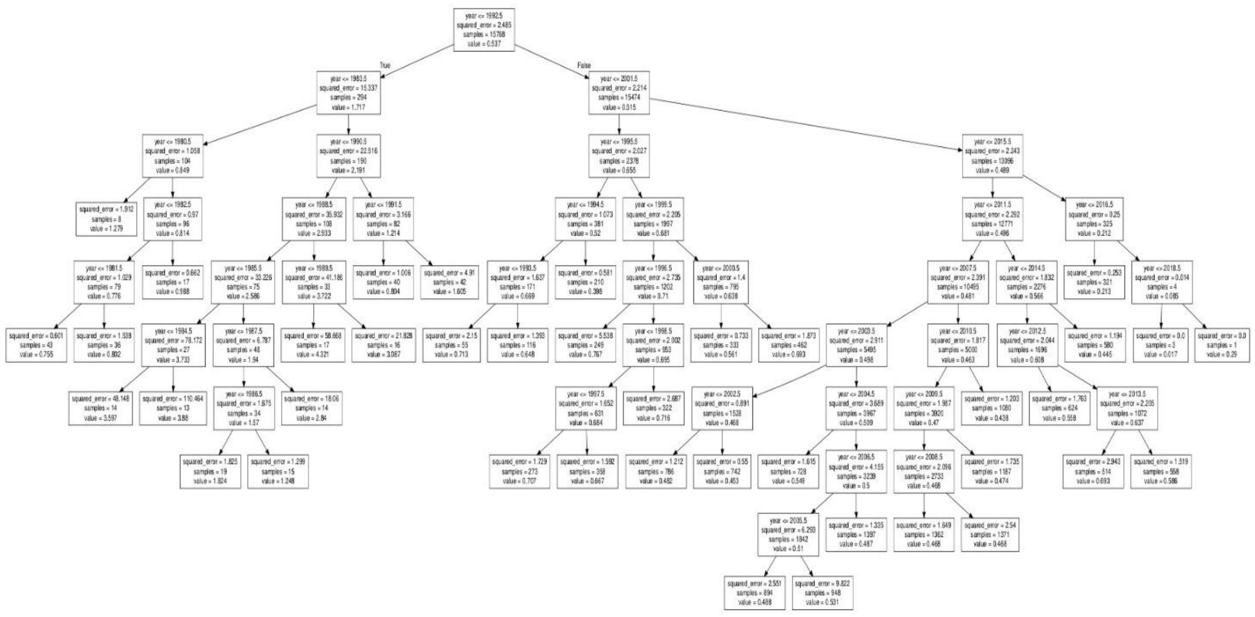
```
X = dataset['Year'].values
y = dataset['Global_Sales'].values

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.05)

# Fitting Decision Tree Regression to the dataset
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor()
regressor.fit(X_train.reshape(-1,1), y_train.reshape(-1,1))

y_pred = regressor.predict(X_test.reshape(-1,1))

df = pd.DataFrame({'Real Values':y_test.reshape(-1), 'Predicted Values':y_pred.reshape(-1)})
df

# Visualising the Decision Tree Regression Results
X_grid = np.arange(min(X), max(X), 0.01)
X_grid = X_grid.reshape((len(X_grid), 1))
plt.scatter(X_test, y_test, color = 'red')
plt.scatter(X_test, y_pred, color = 'green')
plt.title('Decision Tree Regression')
plt.xlabel('year')
plt.ylabel('Global_Sales')
plt.show()

plt.plot(X_grid, regressor.predict(X_grid), color = 'black')
plt.title('Decision Tree Regression')
plt.xlabel('Year')
plt.ylabel('Global_Sales')
plt.show()

# arange for creating a range of values
# from min value of X to max value of X
# with a difference of 0.01 between two
# consecutive values
X_grid = np.arange(min(X), max(X), 0.01)

# reshape for reshaping the data into
# a len(X_grid)*1 array, i.e. to make
# a column out of the X_grid values
X_grid = X_grid.reshape((len(X_grid), 1))

# scatter plot for original data
plt.scatter(X, y, color = 'red')

# plot predicted data
plt.plot(X_grid, regressor.predict(X_grid), color = 'blue')

# specify title
plt.title('Decision Tree Regression')

# specify X axis label
plt.xlabel('year')
```

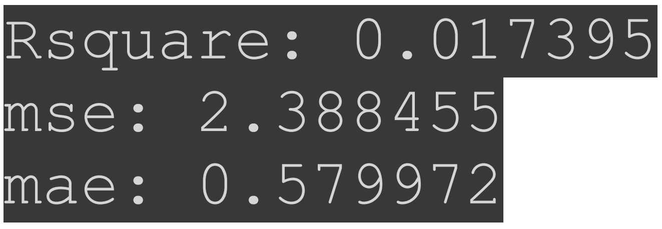
```
# specify Y axis label
plt.ylabel('Global_Sales')

# show the plot
plt.show()

# import export_graphviz
from sklearn.tree import export_graphviz

# export the decision tree to a tree.dot file
# for visualizing the plot easily anywhere
export_graphviz(regressor, out_file ='decisiontree.dot',
            feature_names =['year'])

from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=2529)
X_train.shape,X_test.shape,y_train.shape,y_test.shape

X_test=(X_test.reshape(-1,1))
X_train=(X_train.reshape(-1,1))
y_train=(y_train.reshape(-1,1))
y_test=(y_test.reshape(-1,1))
from sklearn import linear_model
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error,mean_absolute_error
reg_all=DecisionTreeRegressor()
reg_all.fit(X_train,y_train)
y_pred=reg_all.predict(X_test)
Rsquare=reg_all.score(X_test,y_test)
print("Rsquare: %f" %(Rsquare))
#print("Intercept: %f" %(reg_all.intercept_))
mse=mean_squared_error(y_test,y_pred)
print("mse: %f" %(mse))
mae = mean_absolute_error(y_test,y_pred)
print("mae: %f" %(mae))
```











```
Rsquare: 0.017395
mse: 2.388455
mae: 0.579972
```

CHAPTER 5

RESULTS

SUPERVISED MODEL	Error square
LINEAR REGRESSION	0.098596
K NEAREST NEIGHBOUR	2.7498018903614456
SUPORT VECTOR MACHINE(REGRESSOR)	1.019002
DECISION TREE	2.388455

CHAPTER 6

CONCLUSION

So, from the above table(accuracy), we can predict that the linear regression model has shown accurate results for the model. We have used linear regression model to get maximum accuracy rate.

CHAPTER 7

FUTURE SCOPE

In future scope we can develop a artificial intelligence model which can directly get the data from online after release of every game automatically so that there will be no updating of dataset

CHAPTER 8

BIBLIOGRAPHY

1. https://github.com/Jayreddy00/AI-ML-project-SVM-KNN-DECISION-TREE-LINEAR-REGRESSION-

2. https://colab.research.google.com/drive/1Ld6o2zfT6ouQxbjyzzTmBAhdqd4qlt4_#scrollTo=FG7NlJEiHIRQ

3. https://www.kaggle.com/datasets/gregorut/videogamesales