PRASAD V POTLURI SIDDHARTHA
INSTITUTE OF TECHNOLOGY

Thyroid Detection

The thyroid gland plays a crucial role in regulating metabolism, growth, and development.

Presented by:

- **G.HARSHINI (22501A0556)**
- **E.HARI KRISHNA (22501A0545)**
- **G.LAKSHMI PRIYANKA (22501A0546)**
- **Ch. SAI DHEERAJ (22501A0532)**

TABLE OF CONTENTS

- INTRODUCTION
- PROBLEM STATEMENT
- PROCESS TO DO IN PROJECT USING ML
- FLOWCHART
- CODE EXPLANATION
- RESULT OF THE CODE
- CONCLUSION

PROBLEM STATEMENT

- Machine learning models offer the capability to analyze extensive healthcare datasets, uncover exclusive patterns, and detect challenging-to-discern risk factors that conventional methods might miss. These algorithms can predict an individual's Thyroid by examining their health records and lifestyle factors.

- Clinical evaluation includes a detailed medical history and physical examination. Blood tests measure thyroid hormone levels (TSH, Free T4, Free T3).

PROCESS REQUIRED TO DO IN PROJECT USING MACHINE LEARNING

Data Collection

Gather a comprehensive dataset of thyroid-related data for training the machine learning model

Data Pre-Processing

It involves cleaning and transforming raw data into a format that is suitable for analysis or training machine learning models.

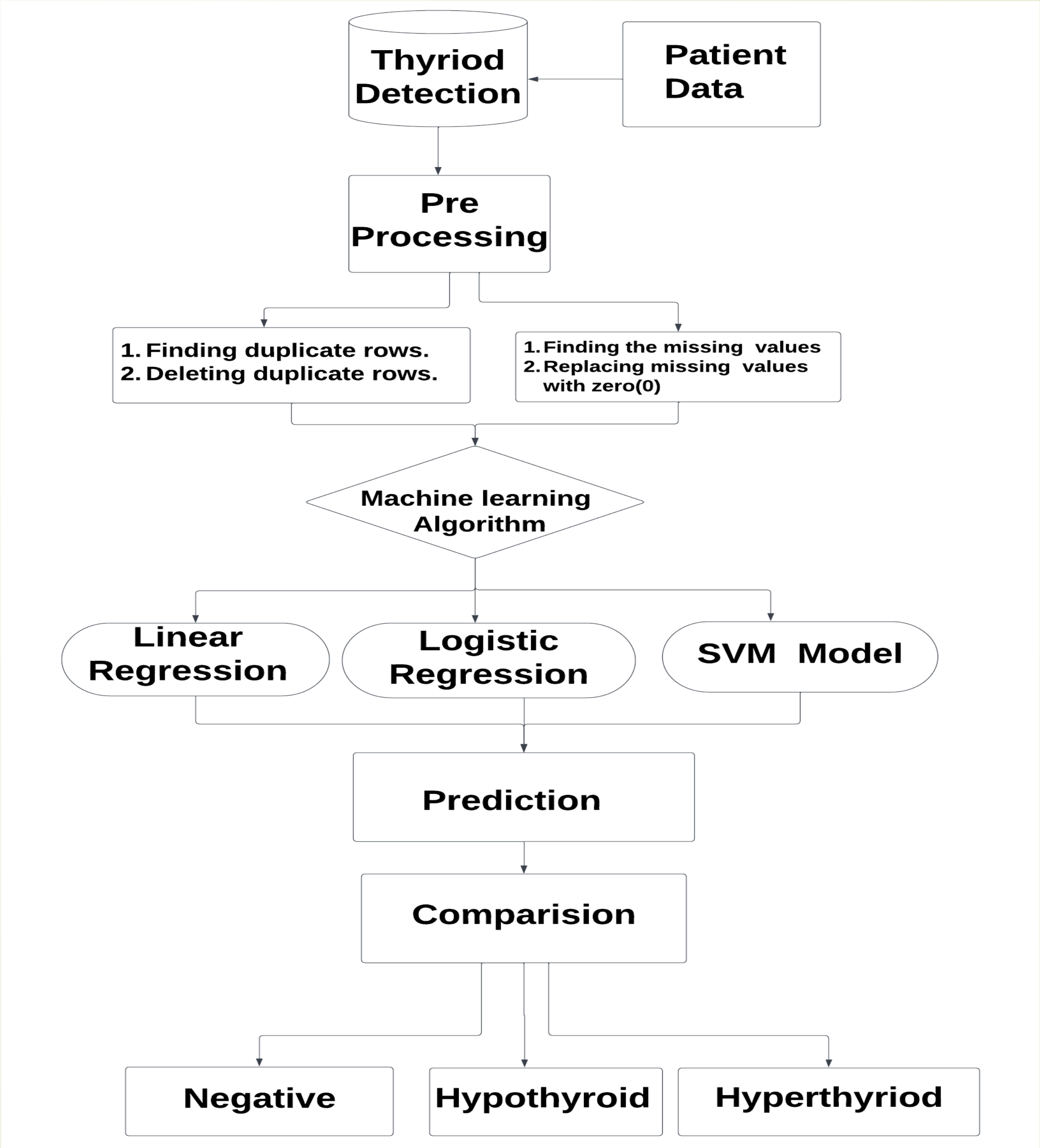
Data Visualization

Data visualization is the graphical representation of data to communicate information, patterns, and insights effectively.

Train and Test Split

one for training the model and another for testing or evaluating the model's performance.

PROJECT FLOWCHART



DATA COLLECTION

1. Data collection is the process of gathering raw information and observations to build a dataset for analysis, research, or other purposes.
2. First import important library functions which are useful to do project.
3. The data is collected from Kaggle app and data is uploaded into the drive and path of the Dataset is copied into the Google Collab.

```
[ ] import numpy as np  # linear algebra
    import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
    import matplotlib.pyplot as plt # deluxe visualization library
    %matplotlib inline
    import seaborn as sns # visualization library to support seaborn
    import warnings # controls warning messages
    import plotly.express as px # creates interactive visualizations and plots(e.g:line charts, bar charts)
    warnings.filterwarnings('ignore') # Filters and ignores warnings, suppressing their display during program execution.
    from sklearn import preprocessing #  Imports the preprocessing module from scikit-learn
```

Read the data from drive

```
[ ] from google.colab import drive
    drive.mount('/content/drive')

    Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

Importing dataset from drive

```
# importing dataset from persistent landing
data=pd.read_csv('/content/drive/MyDrive/thyroidDF.csv')# thyroidDF.csv
```

```
# dataset initial summary
data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9172 entries, 0 to 9171
Data columns (total 23 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   age                  9172 non-null   int64
 1   sex                  8865 non-null   object
 2   on_thyroxine         9172 non-null   object
 3   query_on_thyroxine   9172 non-null   object
 4   on_antithyroid_meds  9172 non-null   object
 5   sick                 9172 non-null   object
 6   pregnant             9172 non-null   object
 7   thyroid_surgery      9172 non-null   object
 8   I131_treatment       9172 non-null   object
 9   query_hypothyroid    9172 non-null   object
 10  query_hyperthyroid   9172 non-null   object
 11  lithium              9172 non-null   object
 12  goitre               9172 non-null   object
 13  tumor                9172 non-null   object
 14  hypopituitary        9172 non-null   object
 15  psych                9172 non-null   object
 16  TSH                  8330 non-null   float64
 17  T3                   6568 non-null   float64
 18  TT4                  8730 non-null   float64
 19  T4U                  8363 non-null   float64
 20  FTI                  8370 non-null   float64
 21  TBG                  349 non-null    float64
 22  target               7679 non-null   object
dtypes: float64(6), int64(1), object(16)
memory usage: 1.6+ MB
```

DATA PRE PROCESSING

- The primary goals of data preprocessing are to address issues such as missing values, outliers, and inconsistencies, and to prepare the data in a way that enhances the performance and interpretability of machine learning models.

CHECKING FOR DUPLICATE DATA

REMOVING DUPLICATE DATA

CHECKING FOR NULL VALUES

FILLING MISSING DATA

DATA VISUALIZATION

CHECK FOR DUPLICATE DATA

1. Finding and handling duplicate values in Python is most important for several reasons like data quality, accuracy etc.
2. For removing duplicates we use the code **data.duplicated()** and it displays duplicated values.
3. Duplicate values can introduce inaccuracies and noise into the dataset. Eliminating duplicates helps maintain a higher level of data quality, ensuring that your analyses and models are based on reliable information.

```
data.duplicated()

0       False
1       False
2       False
3       False
4       False
        ...
9167    False
9168    False
9169    False
9170    False
9171    False
Length: 9172, dtype: bool
```

```
newdata=data.drop_duplicates()

data.isnull().sum()

age                        0
sex                      307
on_thyroxine               0
query_on_thyroxine         0
on_antithyroid_meds        0
sick                       0
pregnant                   0
thyroid_surgery            0
I131_treatment             0
query_hypothyroid          0
query_hyperthyroid         0
lithium                    0
goitre                     0
tumor                      0
hypopituitary              0
psych                      0
TSH_measured               0
TSH                      842
T3_measured                0
T3                      2604
```

CHECK FOR NULL VALUES

- We are Checking for the NaN /Null Values in the Dataset.
- If the Data Contains the Null Values it is not possible to visualize data.
- So we need to fill the Data with ‘0’,’ ffill’, ’bfill’.
- This would also increase the accuracy.

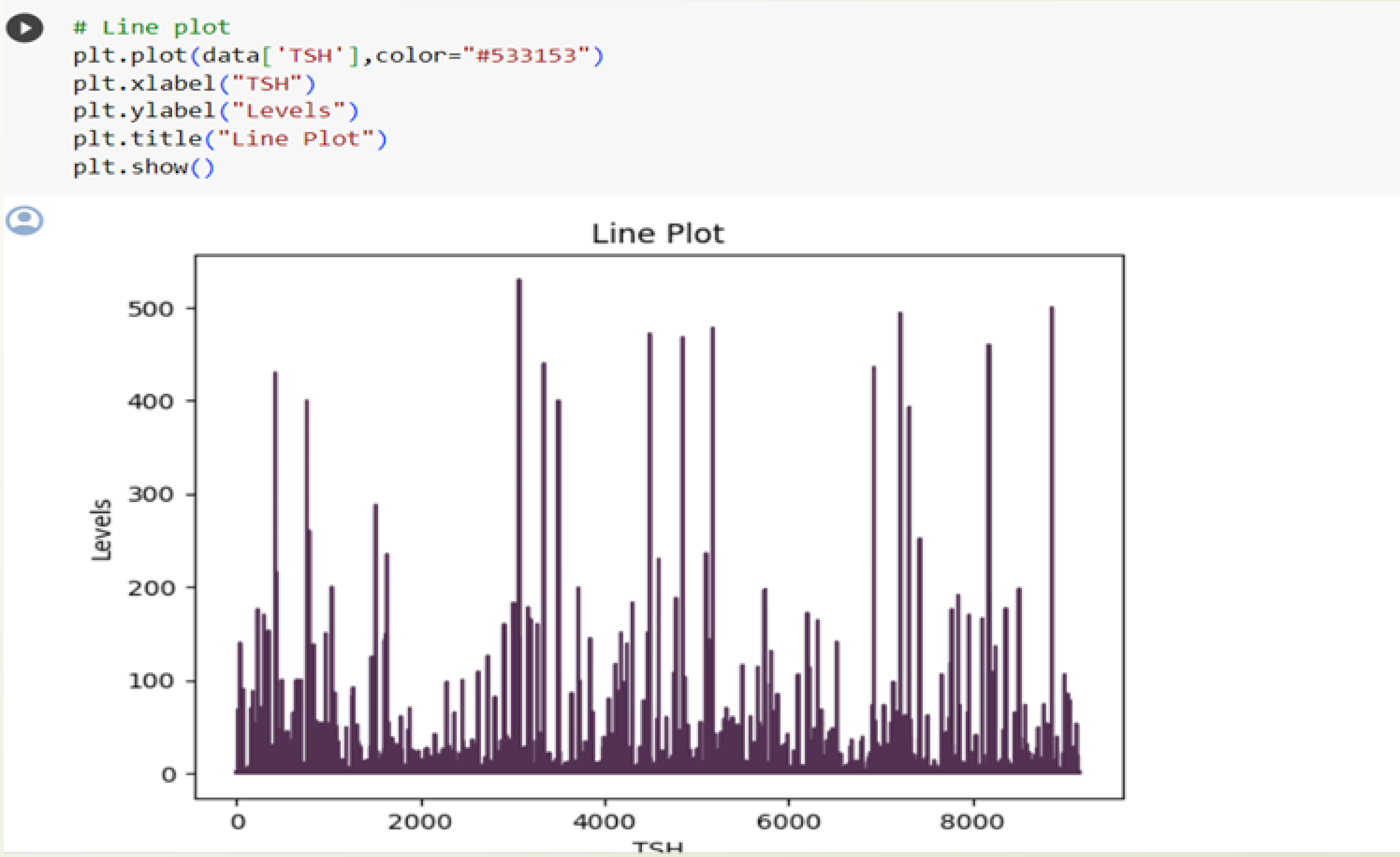
```
data.isnull().sum()# to find null values
```

```
age                       0
sex                     307
on_thyroxine              0
query_on_thyroxine        0
on_antithyroid_meds       0
sick                      0
pregnant                  0
thyroid_surgery           0
I131_treatment            0
query_hypothyroid         0
query_hyperthyroid        0
lithium                   0
goitre                    0
tumor                     0
hypopituitary             0
psych                     0
TSH                     842
T3                     2604
TT4                     442
T4U                     809
FTI                     802
TBG                    8823
target                 1493
dtype: int64
```

```
[ ] # Assuming your DataFrame is 'b_fill_df'
    data['sex'] = data['sex'].fillna(method='ffill')  # Replace missing values in 'sex' column with 0

    # For other columns, you can replace missing values with 0 using similar lines of code:
    data['age'] = data['age'].fillna(method='ffill')
    data['TT4'] = data['TT4'].fillna(method='ffill')
    data['T3'] = data['T3'].fillna(method='ffill')
    data['T4U'] = data['T4U'].fillna(method='ffill')
    data['FTI'] = data['FTI'].fillna(method='ffill')
    data['TSH'] = data['TSH'].fillna(method='ffill')
```

DATA VISUALIZATION

1.Data visualization in Python is crucial for several reasons, especially when working with large and complex datasets.
2.Visualization provides an intuitive way to explore and understand complex datasets. It allows you to identify patterns, trends, and relationships in the data that may not be immediately apparent in raw numbers.
3. Data visualization is the graphical representation of data to communicate information, patterns, and insights effectively.



REPRESENTING THE TARGET IN PIE CHART

```
target_dist = pd.DataFrame(data['target'].value_counts())
target_dist.reset_index(inplace=True)
target_dist.columns = ['target', 'count']
fig = px.pie(target_dist, names='target', values='count')
fig.update_traces(marker=dict(colors=["#BEE9E9", "#6C7B8B", "#3A506B"]))
fig.show()
```

DATA NORMALIZATION

➢ONE HOT ENCODING :

1. One-hot encoding is a technique used in machine learning and data preprocessing to represent categorical variables as binary vectors. It is particularly useful when dealing with categorical data.
2. One-hot encoding is used for ‘input’ variables in the Dataset.

```python
import pandas as pd
from sklearn.preprocessing import OneHotEncoder
categorical_cols = [
 'sex',
 'on_thyroxine',
 'query_on_thyroxine',
 'on_antithyroid_meds',
 'sick',
 'pregnant',
 'thyroid_surgery',
 'I131_treatment',
 'query_hypothyroid',
 'query_hyperthyroid',
 'lithium',
 'goitre',
 'tumor',
 'hypopituitary',
 'psych',
 'TSH',
 'T3',
 'TT4',
 'T4U',
 'FTI',
 'TBG']
encoder = OneHotEncoder(sparse=False, drop='first')  # 'drop' parameter removes one of the one-hot encoded columns to avoid multicollinearity
encoded_cols = pd.DataFrame(encoder.fit_transform(data[categorical_cols]), columns=encoder.get_feature_names_out(categorical_cols))
data = pd.concat([data, encoded_cols], axis=1)
data.drop(categorical_cols, axis=1, inplace=True)
```

```python
from sklearn.impute import SimpleImputer

imputer = SimpleImputer(strategy='most_frequent')
data = pd.DataFrame(imputer.fit_transform(data), columns=data.columns)
```

```python
data
```

➢ LABEL ENCODER:

1. The Label Encoder is a preprocessing tool used in machine learning to convert categorical labels into numerical representations. It is particularly useful when working with algorithms that require numerical input for target variables.
2. Label Encoder is used for ‘output’ variables like “target” int the Dataset.

```
from sklearn.preprocessing import LabelEncoder

# Assuming 'df' is your DataFrame and 'target_column' is the name of your target variable column
target_column = 'target'

# Initialize LabelEncoder
label_encoder = LabelEncoder()

# Fit and transform the target variable
data[target_column] = label_encoder.fit_transform(data[target_column])
```

```
data#checking
```

	age	target	sex_M	on_thyroxine_t	query_on_thyroxine_t	on_antithyroid_meds_t	sick_t	pregnant_t	thyroid_surgery_t	I131_treatment_t	...	TBG_100.0	TBG_106.0	TBG_108.0
0	29.0	2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0
1	29.0	2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0
2	41.0	2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0
3	36.0	2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0
4	32.0	2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9167	56.0	2	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0
9168	22.0	2	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0
9169	69.0	2	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0
9170	47.0	2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0
9171	31.0	2	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0

9172 rows × 1322 columns

TRAIN MODEL

- We use a dataset to train the model using various machine learning algorithms. Training a model is important so that it can understand the various patterns, rules, and features.
- **ML Algorithms:**

1.**Logistic Regression:**

It is like making a smart guess when you have to choose between just two options, like 'yes' or 'no.' It helps you figure out the chances of something happening or not happening.

2.Linear Regression:

Linear regression is a bit like drawing a straight line through points on a graph. Imagine you have some dots on paper, and you want to connect them with a line. Linear regression helps you find the best line that fits those dots.

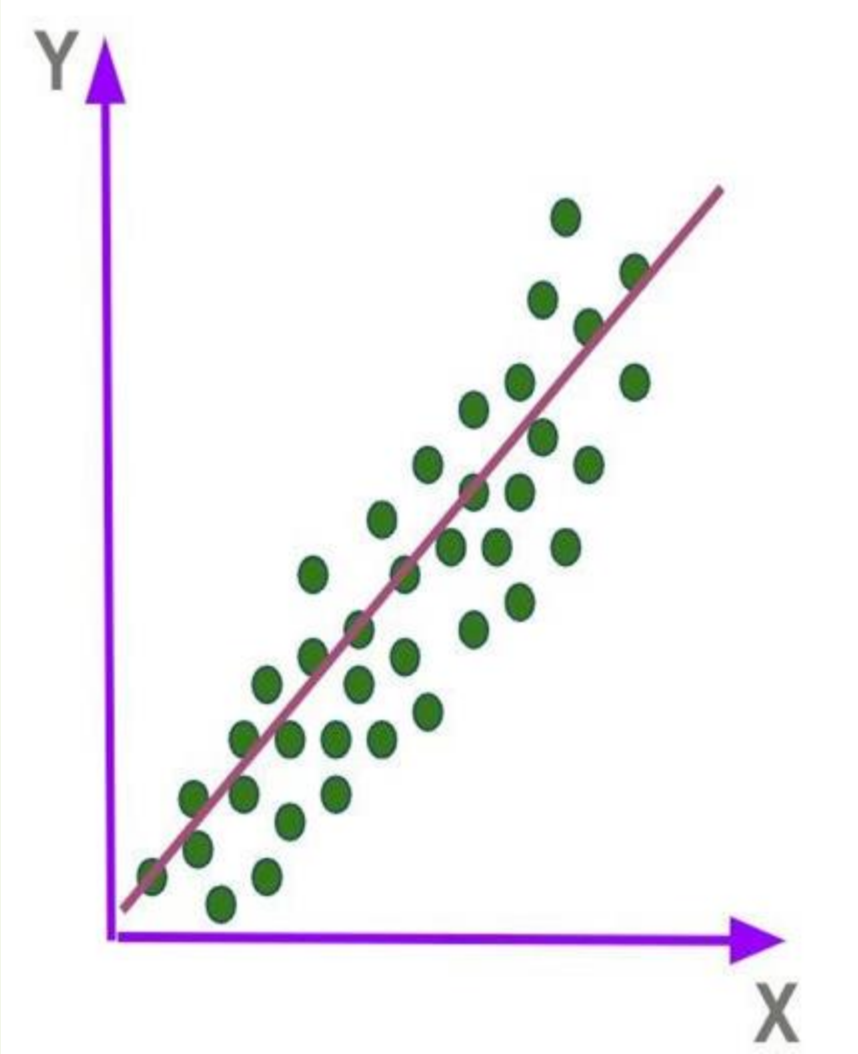
3.SVM Model:

SVM is a versatile supervised learning algorithm used for both classification and regression tasks. The choice of the kernel function in SVM allows it to capture complex relationships in the data, making it effective in various scenarios.

LINEAR REGRESSION

- Linear regression is a bit like drawing a straight line through points on a graph. A Regression analysis is a method for modeling relationships between variables.

```
[ ] from sklearn.linear_model import LinearRegression
    from sklearn.metrics import mean_squared_error, r2_score, accuracy_score
    from sklearn.model_selection import train_test_split
    import numpy as np

    # Assuming X_train, X_test, Y_train, Y_test are your training and testing data

    # Split the data into training and testing sets
    X_train, X_test, Y_train, Y_test = train_test_split(x, y, test_size=0.33, random_state=42)

    # Create a linear regression model
    model = LinearRegression()

    # Fit the model to the training data
    model.fit(X_train, Y_train)

    # Make predictions on the test set
    predictions = model.predict(X_test)

    # Convert predictions to discrete classes (assuming a classification scenario)
    predictions_classes = np.round(predictions).astype(int)

    # Evaluate the model using accuracy (not typical for linear regression)
    accuracy = accuracy_score(Y_test, predictions_classes)

    print('Accuracy of Linear Regression model:', accuracy)

    Accuracy of Linear Regression model: 1.0
```

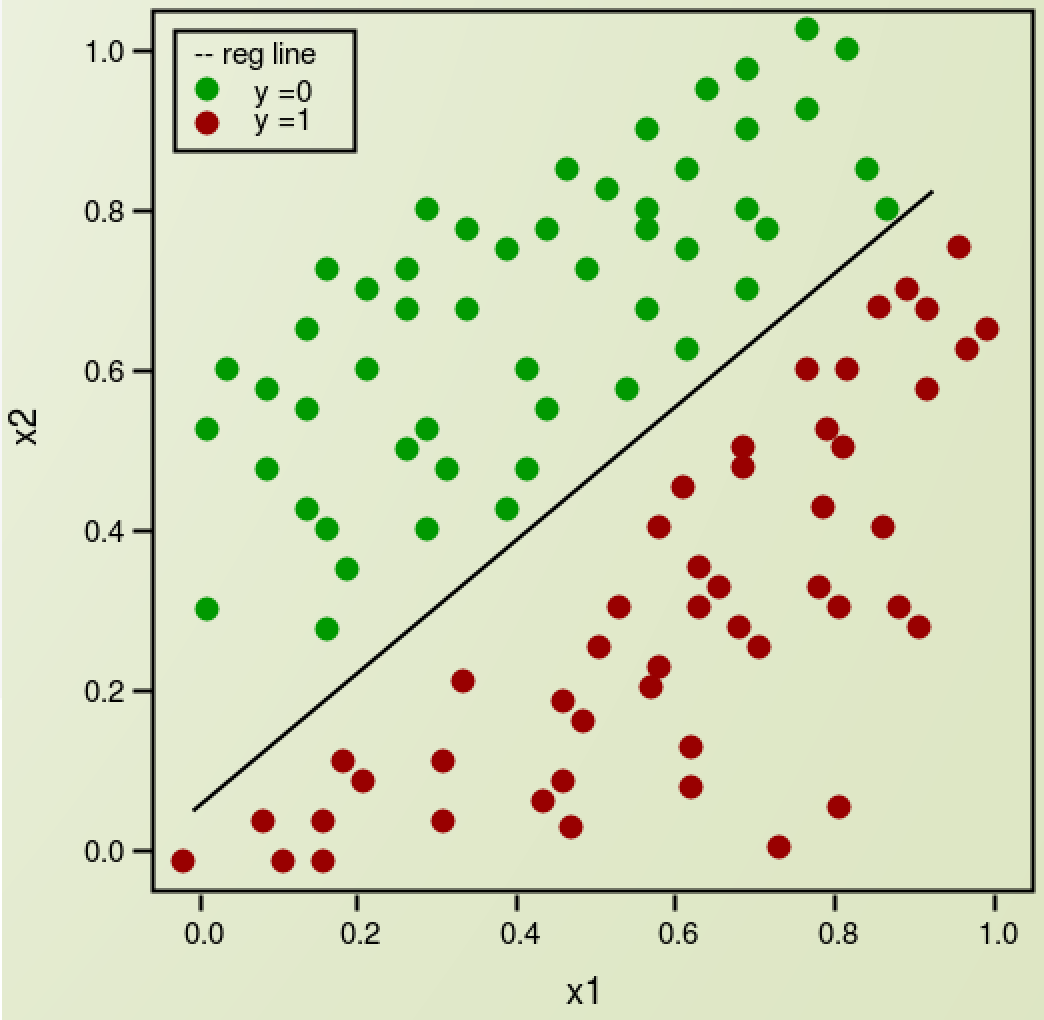


LOGISTIC REGRESSION

- Logistic regression is a statistical method used for binary classification problems, where the dependent variable is categorical and represents two classes (usually 0 and 1). Despite its name, logistic regression is used for classification rather than regression.

```
from sklearn import metrics
from sklearn.metrics import classification_report
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix,ConfusionMatrixDisplay
# Initialize and train the Logistic Regression model
model = LogisticRegression()
model.fit(x_train, y_train)

# Make predictions on the test set
predictions = model.predict(x_test)

# Convert back to original labels for evaluation
predictions_original_labels = label_encoder.inverse_transform(predictions)

# Evaluate the model
accuracy = metrics.accuracy_score(y_test, predictions)
print('The accuracy of the Logistic Regression model is:', accuracy)

# Display the classification report
report = classification_report(y_test, predictions)
print("Classification Report:\n", report)

cm = confusion_matrix(y_test, predictions, labels=model.classes_)
cmap = plt.cm.get_cmap('PuBu')
disp = ConfusionMatrixDisplay(confusion_matrix=cm,display_labels=model.classes_)
disp.plot(cmap =cmap)
plt.show()
```

```
The accuracy of the Logistic Regression model is: 1.0
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        78
           1       1.00      1.00      1.00       209
           2       1.00      1.00      1.00      2740

    accuracy                           1.00      3027
   macro avg       1.00      1.00      1.00      3027
weighted avg       1.00      1.00      1.00      3027
```



SVM MODEL

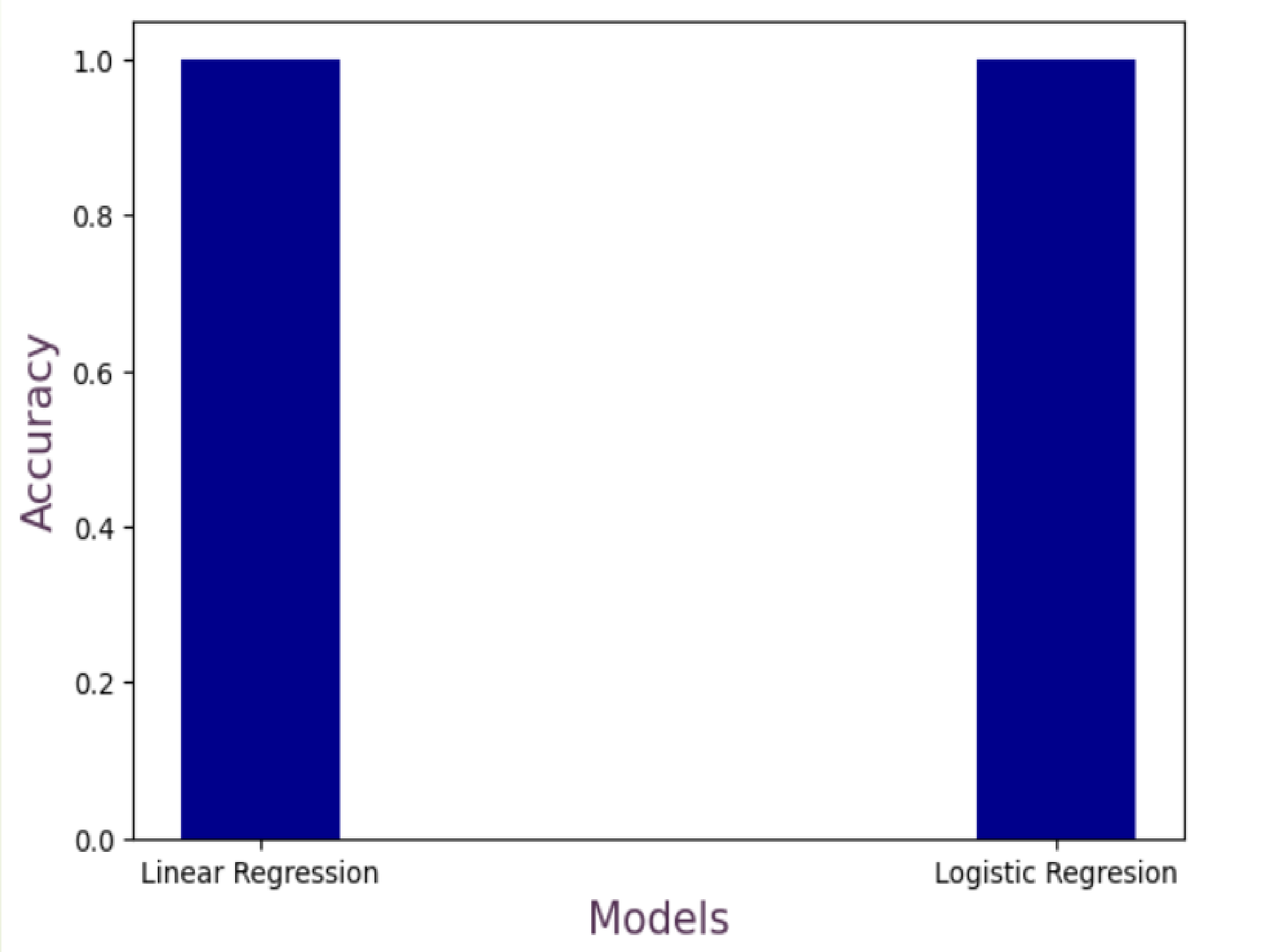
- SVM is used in thyroid detection project alongside linear and logistic regression can enhance the model's ability to handle non-linear relationships, high-dimensional data, and outliers. It provides a overall performance, especially in scenarios where a more complex decision boundary is required.

```
[47] from sklearn import metrics
     from sklearn.metrics import classification_report
     from sklearn.preprocessing import LabelEncoder
     from sklearn.svm import SVC


     # Initialize and train the Logistic Regression model
     model = SVC()
     model.fit(x_train, y_train)

     # Make predictions on the test set
     predictions = model.predict(x_test)

     # Convert back to original labels for evaluation
     predictions_original_labels = label_encoder.inverse_transform(predictions)

     # Evaluate the model
     accuracy = metrics.accuracy_score(y_test, predictions)
     print('The accuracy of the svm model is:', accuracy)

     # Display the classification report
     report = classification_report(y_test, predictions)
     print("Classification Report:\n", report)
```

```
The accuracy of the svm model is: 0.905186653452263
Classification Report:
               precision    recall  f1-score   support

           0       0.00      0.00      0.00        78
           1       0.00      0.00      0.00       209
           2       0.91      1.00      0.95      2740

    accuracy                           0.91      3027
   macro avg       0.30      0.33      0.32      3027
weighted avg       0.82      0.91      0.86      3027
```

Comparison between Linear and Logistic Regression

The following plot is a comparison between accuracies of Linear and Logistic Regression.

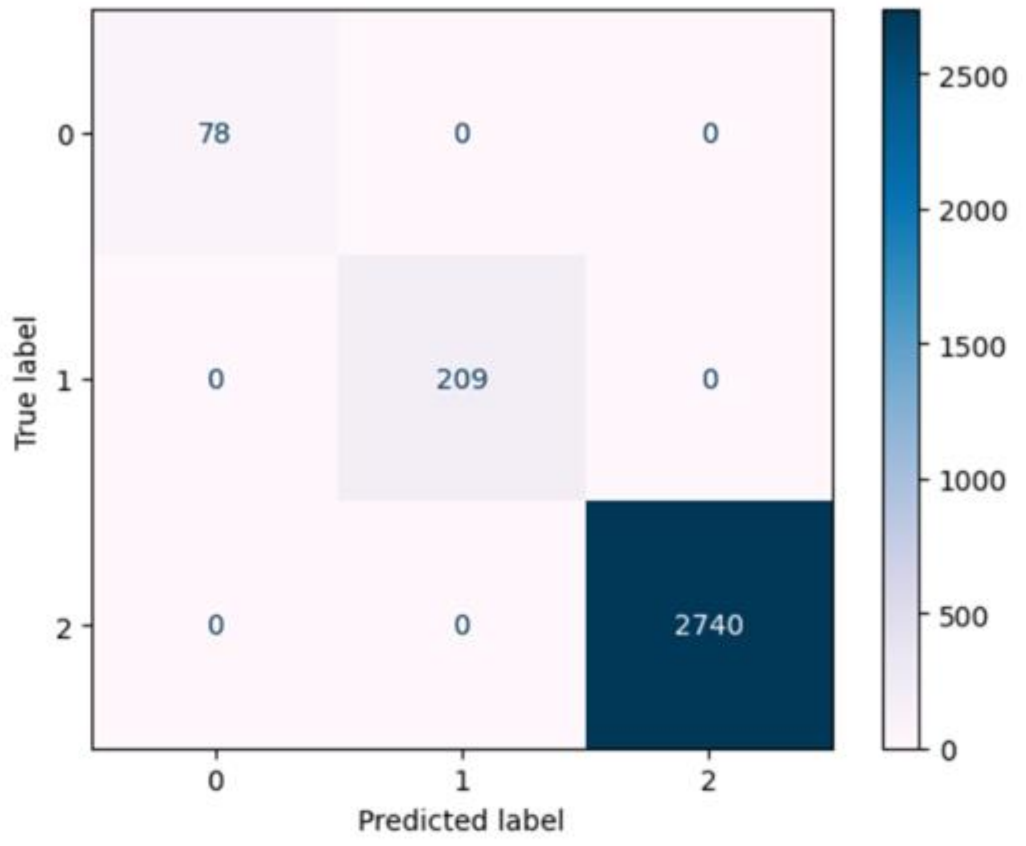


RESULT

ACCURACY FOR LOGISTIC REGRESSION:

The accuracy of the Logistic Regression model is: 1.0
Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	78
1	1.00	1.00	1.00	209
2	1.00	1.00	1.00	2740
accuracy			1.00	3027
macro avg	1.00	1.00	1.00	3027
weighted avg	1.00	1.00	1.00	3027



ACCURACY FOR LINEAR REGRESSION:

Accuracy of Linear Regression model: 1.0

ACCURACY FOR SVM MODEL:

The accuracy of the svm model is: 0.905186653452263
Classification Report:

	precision	recall	f1-score	support
0	0.00	0.00	0.00	78
1	0.00	0.00	0.00	209
2	0.91	1.00	0.95	2740
accuracy			0.91	3027
macro avg	0.30	0.33	0.32	3027
weighted avg	0.82	0.91	0.86	3027

CONCLUSION

- The thyroid detection project aimed to develop a predictive model to determine whether an individual has thyroid-related conditions based on a given dataset.

- In conclusion, the thyroid detection project demonstrated the potential of machine learning techniques in predicting thyroid conditions based on patient data. The project's success relied on a holistic approach that integrated domain knowledge, data preprocessing, model development, and careful evaluation.

THANK YOU