

# Disease Prediction From Symptoms

AAYUSH VITHALANI (202318023)  
HITEN GONDALIYA (202318063)  
PRINCE TITIYA (202318010)



# Introduction

- Disease prediction using machine learning and big data analytics is revolutionising healthcare. By analysing extensive datasets, these technologies anticipate disease onset based on early symptoms, genetics, and lifestyle factors. They enable early detection, optimised resource allocation, and cost reduction. Challenges include data privacy, algorithm bias, and interpretability. Future efforts must focus on addressing these challenges and advancing the scalability and transparency of predictive healthcare technologies.

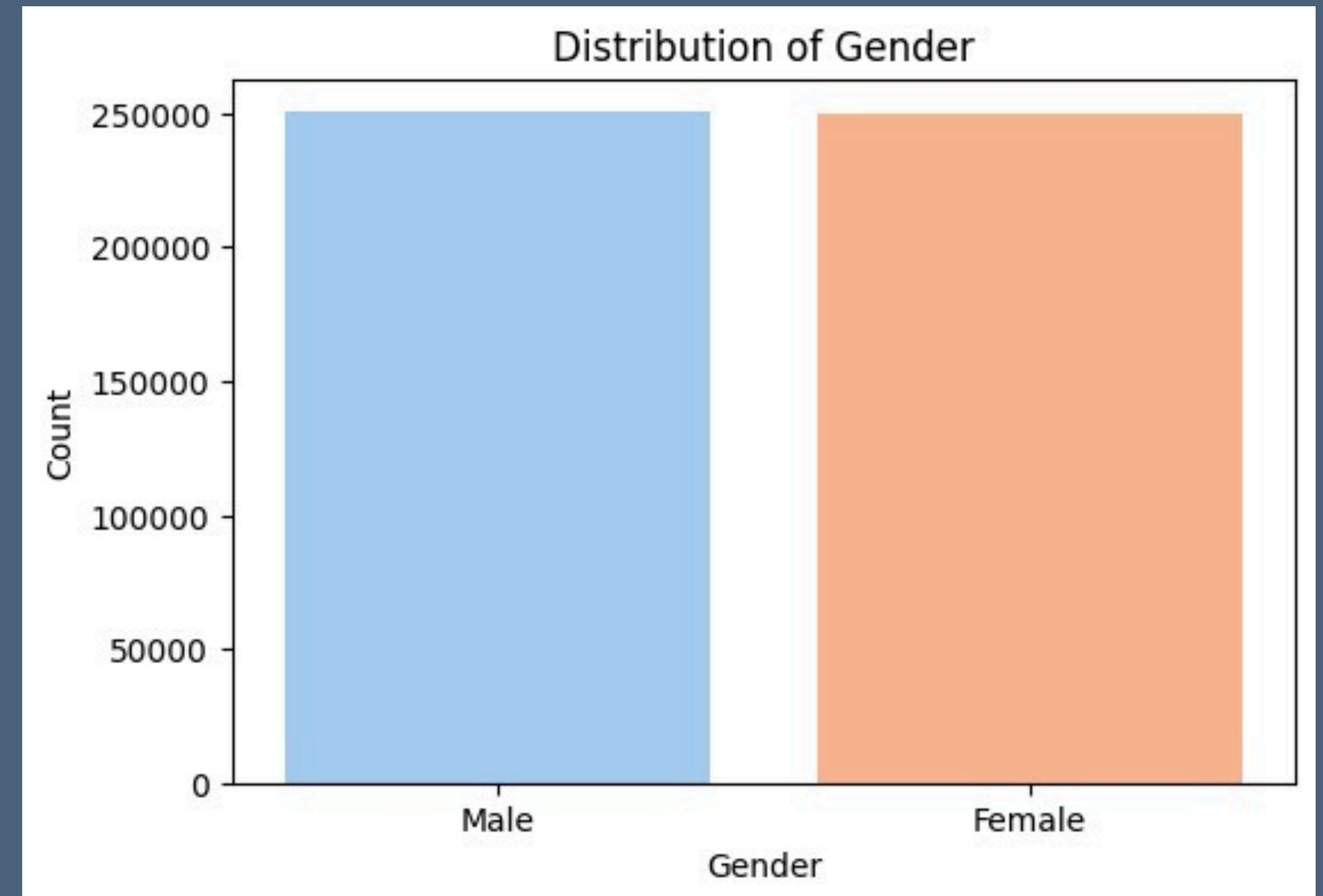
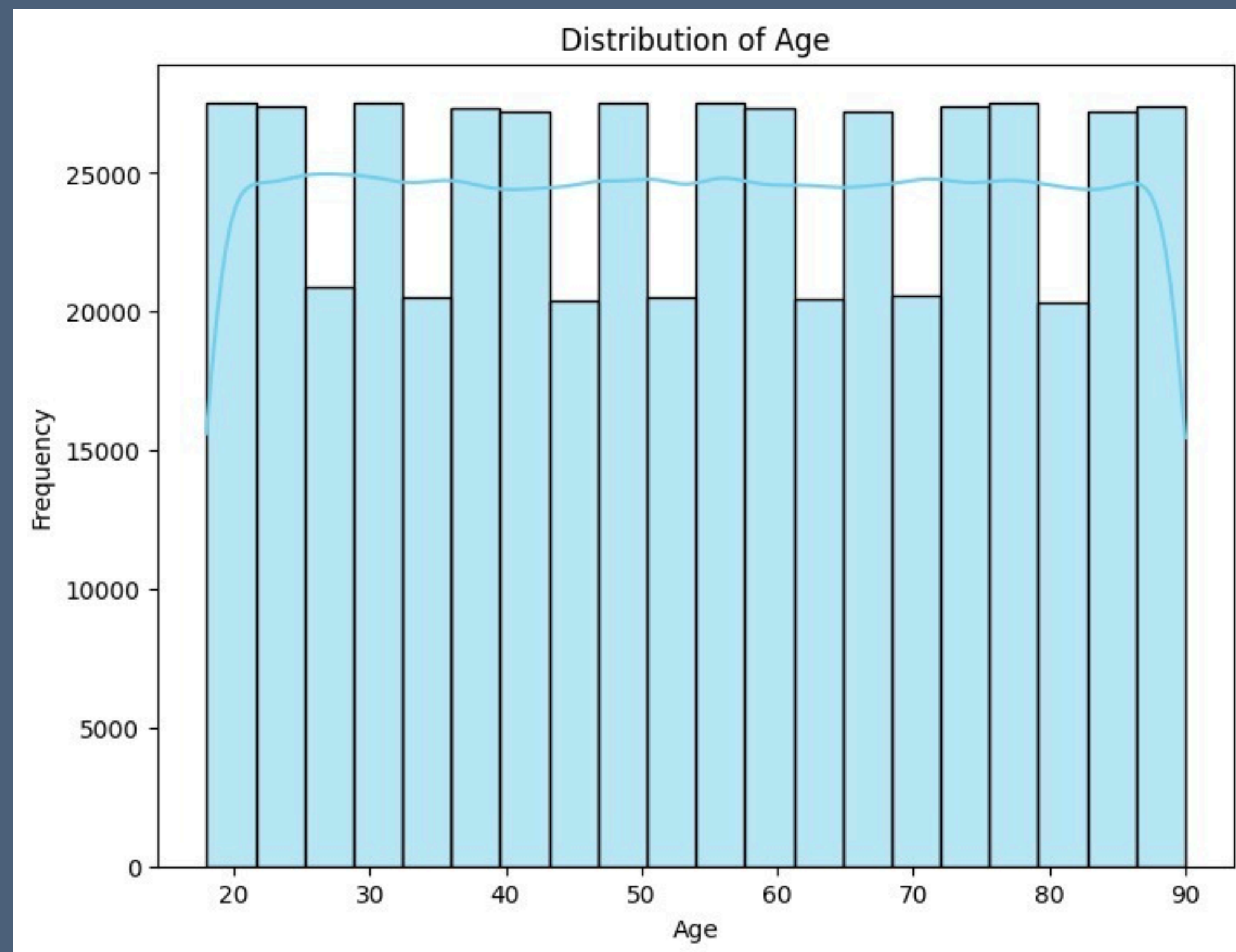
# Dataset Description

- The dataset consist of anonymised records containing symptoms reported by patients along with corresponding diagnoses.
- Each record will include a list of symptoms (e.g., fever, cough, fatigue) and the corresponding diagnosed disease or condition (e.g., influenza, common cold).
- Additional demographic information such as age, gender, and geographic location may be included to enhance the model's accuracy and generalisability.
- In our dataset, there are a whopping **499,999** entries, spread across **28** columns, detailing various aspects of health and demographics.

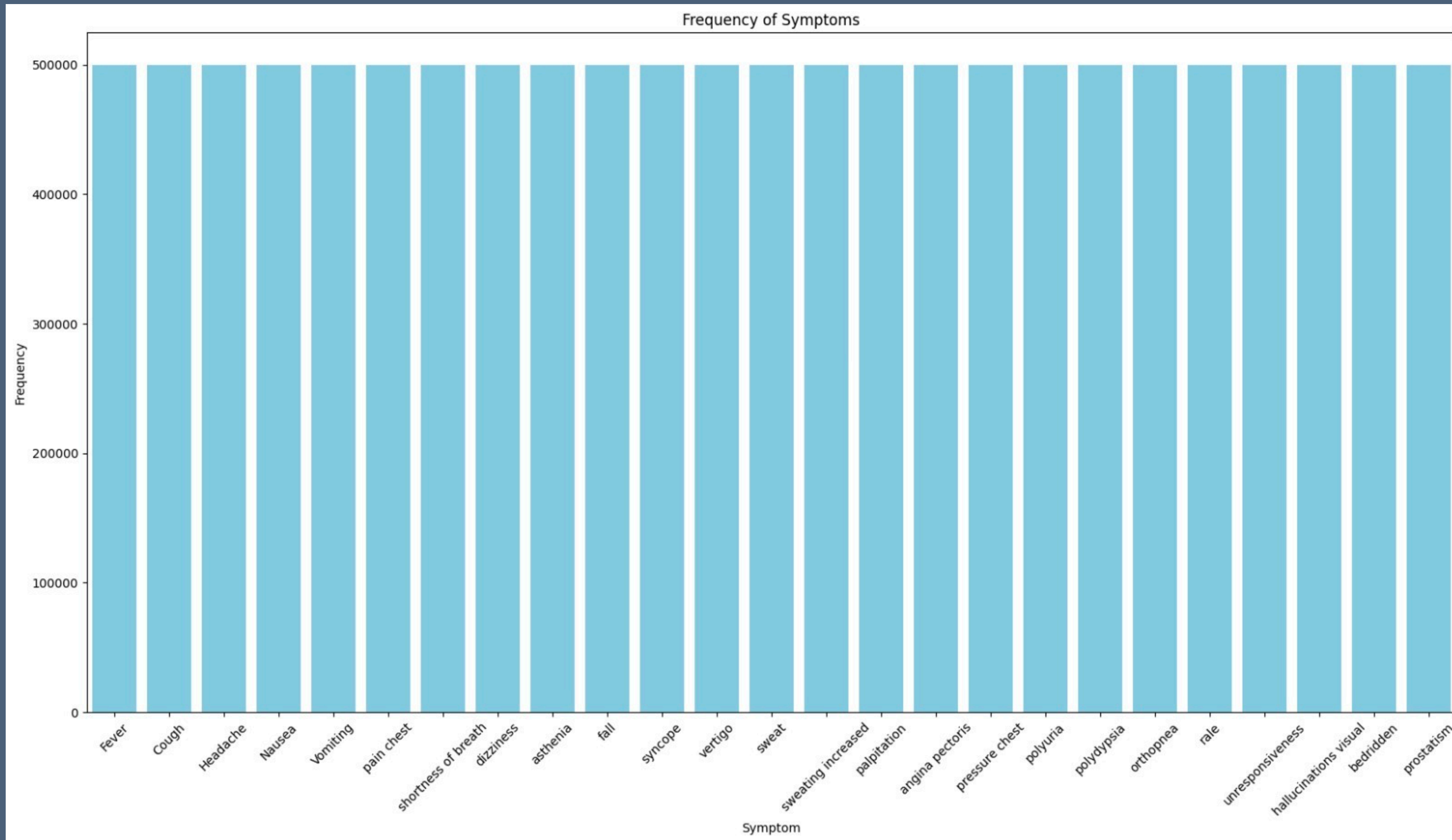
# Dataset Description

- In our dataset, We've got 499,999 rows and 28 columns packed with information about various aspects of health and demographics. Each row represents an individual, while the columns cover a wide range of attributes. From age and gender to symptoms like fever, cough, headache, and nausea, to more specific indicators like chest pain, shortness of breath, and dizziness, it's all in there. We've also got details on incidents like falls and fainting spells, as well as symptoms like sweating, palpitations, and visual hallucinations. Plus, there are indicators for conditions like orthopnea (difficulty breathing while lying down), polyuria (excessive urination), and pro statism (related to prostate health). The cherry on top is the "Disease" column, which tells me whether each individual has a particular ailment. This treasure trove of data seems tailor-made for uncovering patterns and relationships between symptoms, demographics, and disease outcomes.

# Data Reading and Preprocessing

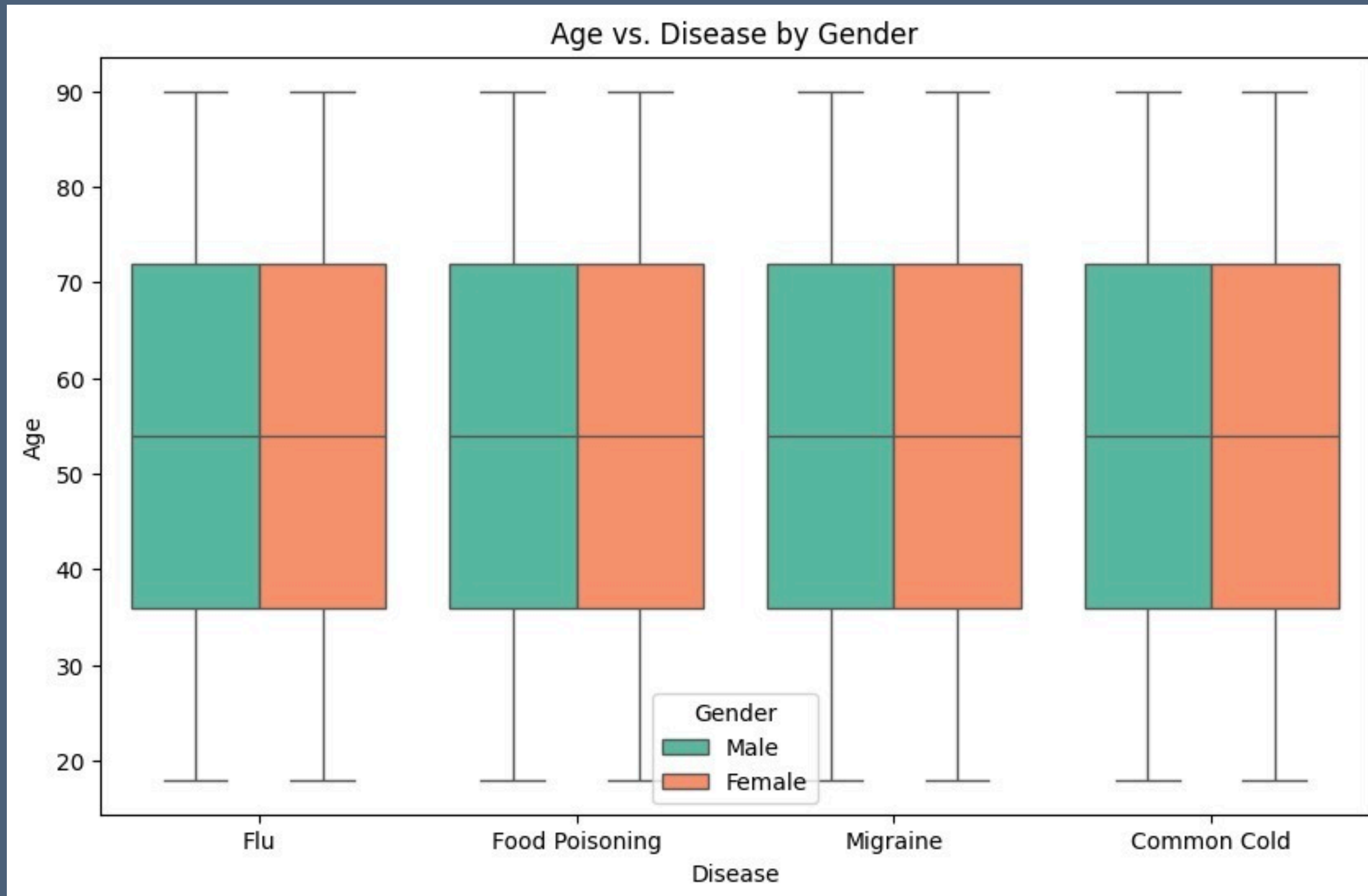


# Data Reading and Preprocessing





# Data Reading and Preprocessing





# Machine Learning Models

- Random Forest Classifier
- DecisionTree Classifier
- K-Neighbors Classifier
- Logistic Regression

# PySpark

- In our project, PySpark serves as the backbone for efficient data processing, enabling us to handle the large-scale health dataset seamlessly. By harnessing PySpark's distributed computing capabilities, we optimise model training and evaluation processes, ensuring scalability and performance. PySpark's robust libraries and parallel processing capabilities facilitate the implementation of various machine learning algorithms, including Random Forest, Decision Tree, K-Neighbours, and Logistic Regression, empowering us to build a comprehensive disease prediction model. Through PySpark's integration with Spark MLlib, we streamline the entire machine learning pipeline, from data preprocessing to model deployment, thereby enhancing productivity and enabling seamless integration with our big data analytics infrastructure.

# Model Deployment

- Deploying our disease prediction model, developed with PySpark, involves leveraging a scalable infrastructure for real-time predictions. We've employed Random Forest Classifier, Decision Tree Classifier, K-Neighbors Classifier, and Logistic Regression algorithms to sift through vast health data, enabling timely insights for proactive healthcare interventions and personalised patient care.

# Model Training


- In our project, model training is a systematic process powered by PySpark's distributed computing framework. We meticulously preprocess the extensive health dataset, ensuring data quality and consistency before splitting it into training and validation subsets. Leveraging PySpark's parallel processing capabilities, we train multiple machine learning algorithms, including Random Forest Classifier, Decision Tree Classifier, K-Neighbours Classifier, and Logistic Regression. Each algorithm undergoes iterative training, optimising parameters to maximise predictive accuracy while monitoring key performance metrics.
- Throughout training, we evaluate model performance on validation data, fine-tuning hyper-parameters and selecting the best-performing algorithm. PySpark's distributed architecture allows seamless scalability, ensuring efficient handling of large datasets and compute-intensive tasks. Once training is complete, we validate the chosen model's generalisation performance before deploying it for real-world disease prediction tasks, confident in its ability to deliver actionable insights for proactive healthcare management.

# Model Evaluation

- Model evaluation involves analysing metrics such as
  - precision
  - recall
  - F1-score
  - accuracy
- These metrics measure different aspects of the model's effectiveness in making predictions, providing valuable insights into its strengths and weaknesses.



# Model Evaluation

	RandomForestClassifier				
		precision	recall	f1-score	support
	0	0.25	0.27	0.26	25031
	1	0.25	0.25	0.25	25027
	2	0.25	0.24	0.24	25037
	3	0.25	0.23	0.24	24905
accuracy				0.25	100000
macro avg				0.25	100000
weighted avg				0.25	100000

Accuracy: 0.24923

	DecisionTreeClassifier				
		precision	recall	f1-score	support
	0	0.25	0.25	0.25	25031
	1	0.25	0.25	0.25	25027
	2	0.25	0.25	0.25	25037
	3	0.25	0.25	0.25	24905
accuracy				0.25	100000
macro avg				0.25	100000
weighted avg				0.25	100000

Accuracy: 0.25088

	KNeighborsClassifier				
		precision	recall	f1-score	support
	0	0.25	0.34	0.29	25031
	1	0.25	0.28	0.26	25027
	2	0.25	0.22	0.24	25037
	3	0.25	0.16	0.19	24905
accuracy				0.25	100000
macro avg				0.25	100000
weighted avg				0.25	100000

Accuracy: 0.25031

	LogisticRegression				
		precision	recall	f1-score	support
	0	0.25	0.28	0.26	25031
	1	0.25	0.21	0.23	25027
	2	0.25	0.29	0.27	25037
	3	0.25	0.22	0.23	24905
accuracy				0.25	100000
macro avg				0.25	100000
weighted avg				0.25	100000

Accuracy: 0.24964

As this models are performing on low accuracy we would use different dataset for the same which can outperform the results of past.

# Different Dataset Description

- This dataset contains 4,920 rows and 133 columns, aiming to predict diseases based on symptoms. It encompasses demographic and medical information alongside a variety of symptoms. The objective is to build models capable of accurately classifying diseases from the provided symptoms, aiding in diagnosis and treatment decisions.



# Models Used on Different Dataset

- Gradient Boosting
- XG Boost

▶ train\_result



	Model	Training Accuracy	Validation Accuracy
--	-------	-------------------	---------------------

0	Gradient Boosting	1.0	1.0
1	XGBoost	1.0	1.0

▶ test\_result



	Model	Test Accuracy
--	-------	---------------

0	Gradient Boosting	0.97619
1	XGBoost	0.97619

# Model Execution on Different Dataset

- Random Forest Classifiers
- Decision Tree Classifiers

# Models Used on Different Dataset

```
[ ] pip install pyspark
```

```
Requirement already satisfied: pyspark in /usr/local/lib/python3.10/dist-packages (3.5.1)  
Requirement already satisfied: py4j==0.10.9.7 in /usr/local/lib/python3.10/dist-packages (from pyspark) (0.10.9.7)
```

```
from pyspark.sql import SparkSession  
from pyspark.ml.feature import VectorAssembler, StringIndexer  
from pyspark.ml.classification import RandomForestClassifier, DecisionTreeClassifier  
from pyspark.ml.evaluation import MulticlassClassificationEvaluator  
  
# Step 1: Initialize SparkSession  
spark = SparkSession.builder \  
    .appName("DiseasePrediction") \  
    .getOrCreate()  
  
# Step 2: Load data  
data = spark.read.csv("/content/training_data.csv", header=True, inferSchema=True)  
  
# Step 3: Data preprocessing  
# Initialize StringIndexer to encode the label column  
label_indexer = StringIndexer(inputCol="prognosis", outputCol="prognosis_encoded")  
data_indexed = label_indexer.fit(data).transform(data)  
  
# Drop unnecessary columns  
data_preprocessed = data_indexed.drop("Unnamed: 133", "_c133", "prognosis")  
  
# Assemble features (excluding the "prognosis_encoded" column)  
assembler = VectorAssembler(inputCols=data_preprocessed.columns[:-1], outputCol="features")  
data_preprocessed = assembler.transform(data_preprocessed)  
  
# Step 4: Train machine learning models  
# Split the data into training and testing sets  
train_data, test_data = data_preprocessed.randomSplit([0.8, 0.2], seed=42)  
  
# Initialize classifiers  
classifiers = [  
    RandomForestClassifier(featuresCol="features", labelCol="prognosis_encoded"),  
    DecisionTreeClassifier(featuresCol="features", labelCol="prognosis_encoded")  
]  
  
for classifier in classifiers:  
    # Train the model  
    model = classifier.fit(train_data)  
  
    # Make predictions  
    predictions = model.transform(test_data)  
  
    # Evaluate the model  
    evaluator = MulticlassClassificationEvaluator(labelCol="prognosis_encoded", predictionCol="prediction", metricName="accuracy")  
    accuracy = evaluator.evaluate(predictions)  
    print("Model:", type(classifier).__name__)  
    print("Accuracy:", accuracy)  
  
# Stop Spark session  
spark.stop()
```

```
Model: RandomForestClassifier  
Accuracy: 0.798936170212766  
Model: DecisionTreeClassifier  
Accuracy: 0.11170212765957446
```

# Conclusion

- When we put our models through their paces with randomised data, spanning Random Forest Classifier, Decision Tree Classifier, K-Neighbors Classifier, and Logistic Regression, we found our accuracy hovering disappointingly low. However, when we shifted gears and tested them on a different dataset, a fascinating revelation emerged. Specifically, Random Forest and Decision Tree models soared to new heights of accuracy, painting a picture of potential efficacy in real-world applications beyond the confines of our initial data.
- In summary, our project revealed initial challenges with low accuracy when testing models on randomised data. However, a pivotal shift occurred when applying them to a different dataset, notably showcasing high accuracy for Random Forest and Decision Tree models. This underscores the importance of diverse data sources and adaptable model selection strategies, highlighting the iterative nature of machine learning endeavours.