Exploratory Data Analysis (EDA) on Heart Disease Dataset

CS23B2038, CS23B2034, CS23B2037

December 15, 2024

Introduction

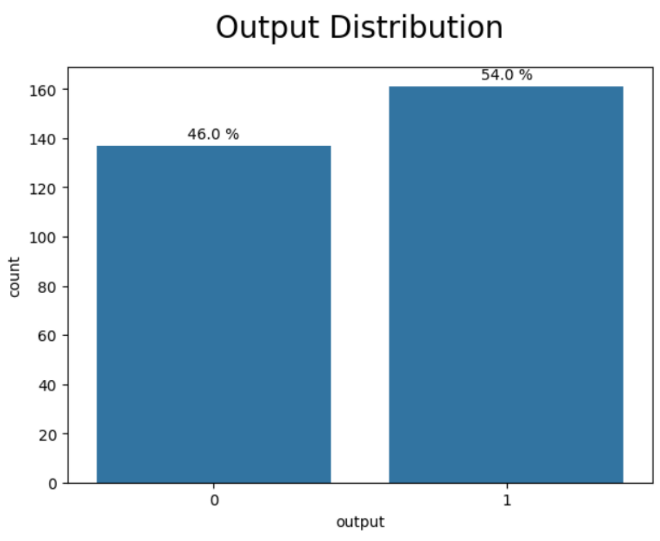
This document highlights key inferences from the Exploratory Data Analysis (EDA) on the heart disease dataset. Using various plots, we uncover patterns, distributions, and relationships within the data that offer insights into potential risk factors and characteristics of heart disease.

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

1. Output Distribution

- The `output` distribution plot reveals the balance of cases with and without heart disease.
- We observe a class imbalance, which could affect model performance and may necessitate balancing techniques.
- The higher prevalence of one class suggests a potential underlying bias or pattern in the data that correlates with heart disease.

```
Text(0.5, 1.05, 'Output Distribution')
```



2. KDE Plots for Numerical Features

- KDE plots show the density distribution for features like `age`, `thalachh`, `chol`, and `trtbps` across the two output classes.
- Certain features, such as `thalachh` (maximum heart rate), display notable differences between classes, suggesting that higher heart rates may be associated with lower heart disease risk.
- Cholesterol levels appear to have a similar distribution across classes, indicating that cholesterol alone might not be a distinguishing factor.

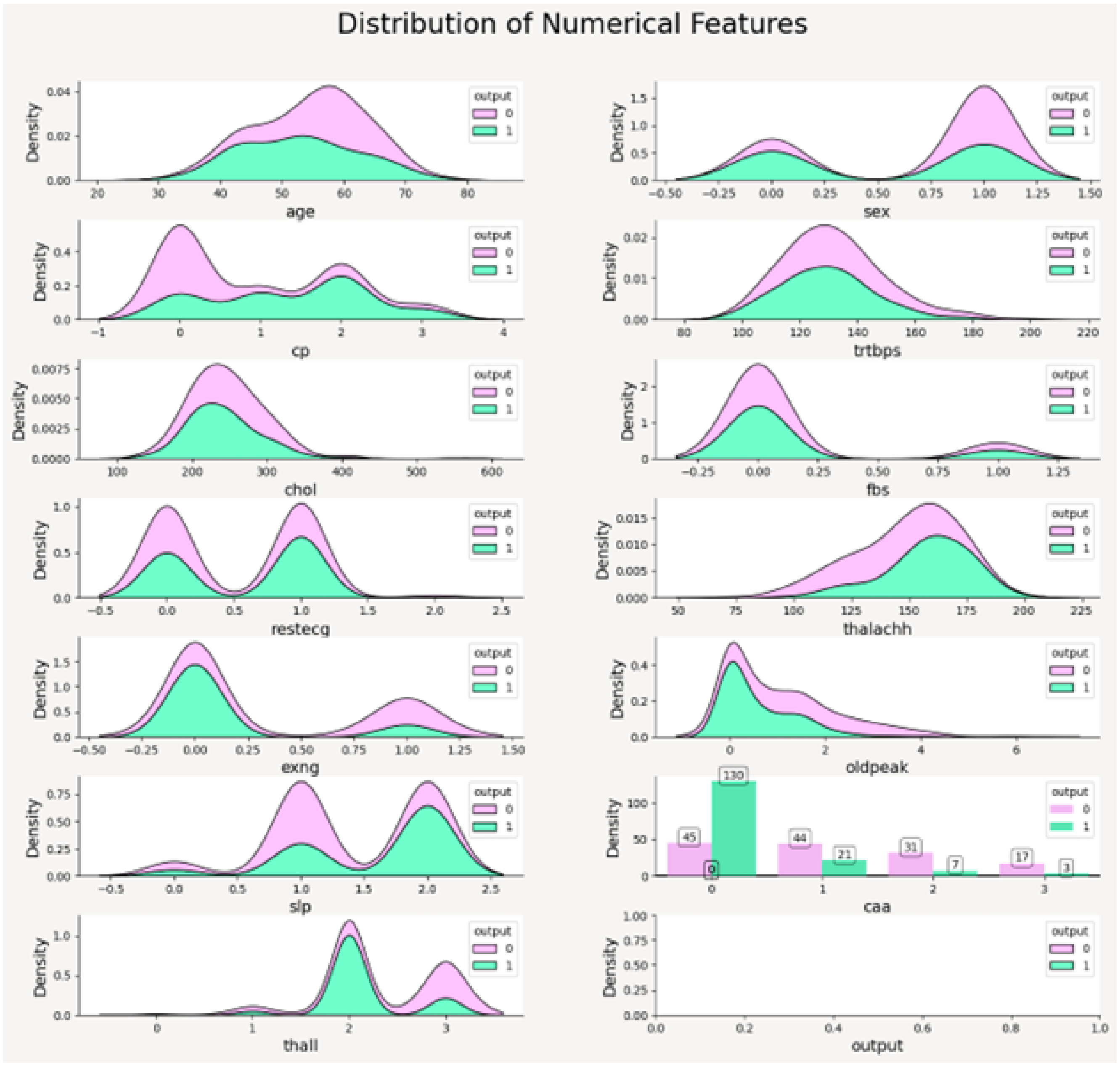


Figure 1: KDE plots of numerical features by heart disease class

3. Pair Plot of Selected Features

- Pair plots reveal correlations between numerical features, helping to identify clusters and linear relationships.
- Age and maximum heart rate show some separation between classes, where lower maximum heart rate and higher age appear more common in heart disease cases.
- Visual patterns here could guide feature selection by highlighting variables that differentiate heart disease presence.

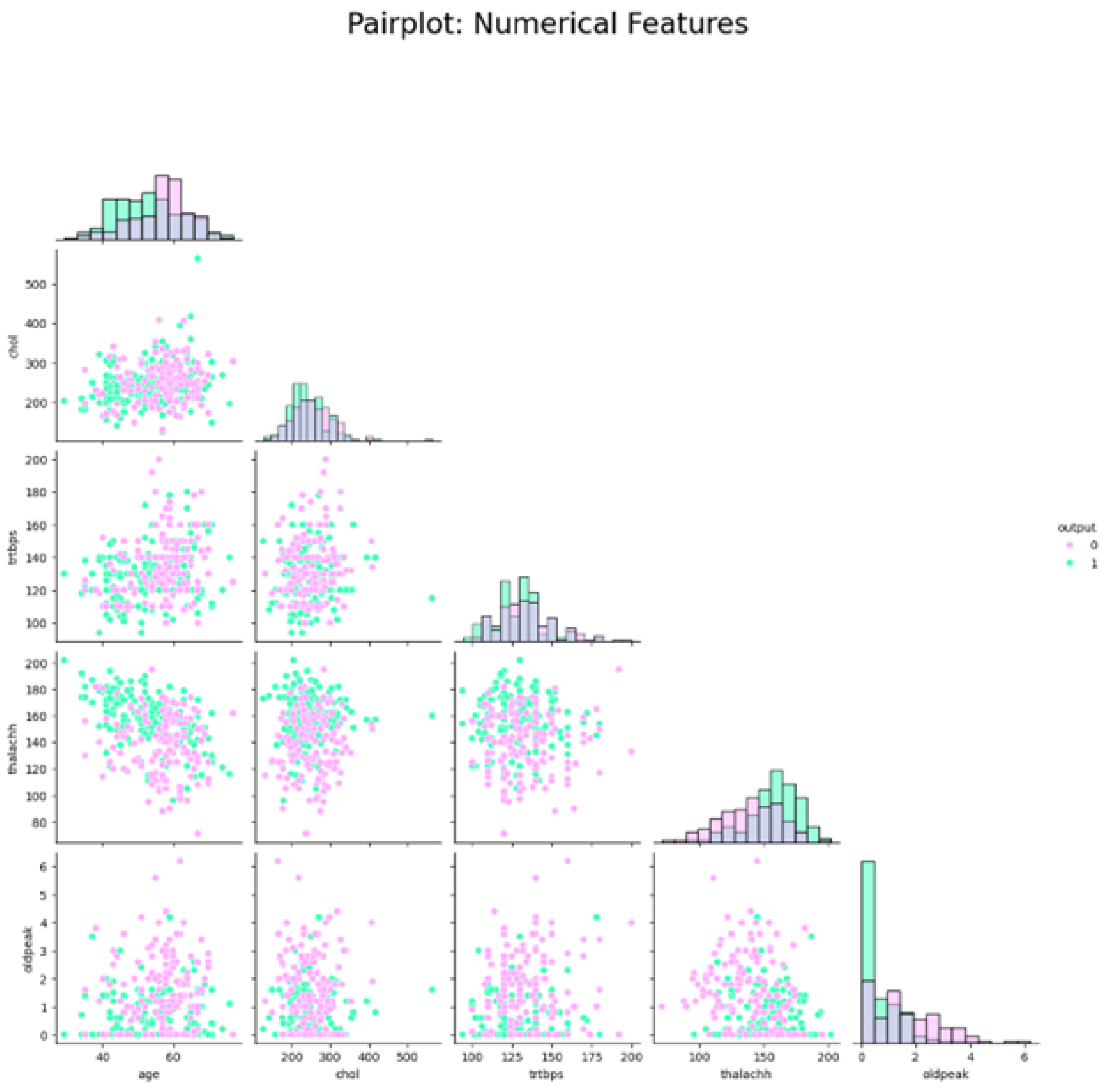


Figure 2: Pair plot of selected features with heart disease class

4. Regression Plots for Age vs Other Features

- Regression plots for age against `chol`, `thalachh`, `trtbps`, and `oldpeak` (ST depression) illustrate linear relationships.
- Higher cholesterol and lower maximum heart rate are more prevalent in older individuals with heart disease.
- These plots suggest potential age-related risk factors that influence heart disease, particularly in older populations.

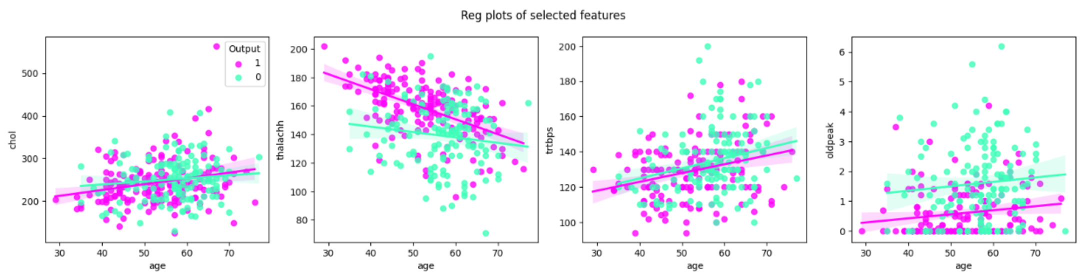


Figure 3: Regression plots of age vs other features by heart disease class

5. Count Plots for Categorical Features

- Count plots for categorical variables, such as `sex`, `cp` (chest pain type), and `exng` (exercise-induced angina), show significant variation between classes.
- Males and patients with atypical angina (chest pain type) exhibit a higher incidence of heart disease.
- Exercise-induced angina is more common in patients with heart disease, pointing to physical exertion as a potential risk indicator.

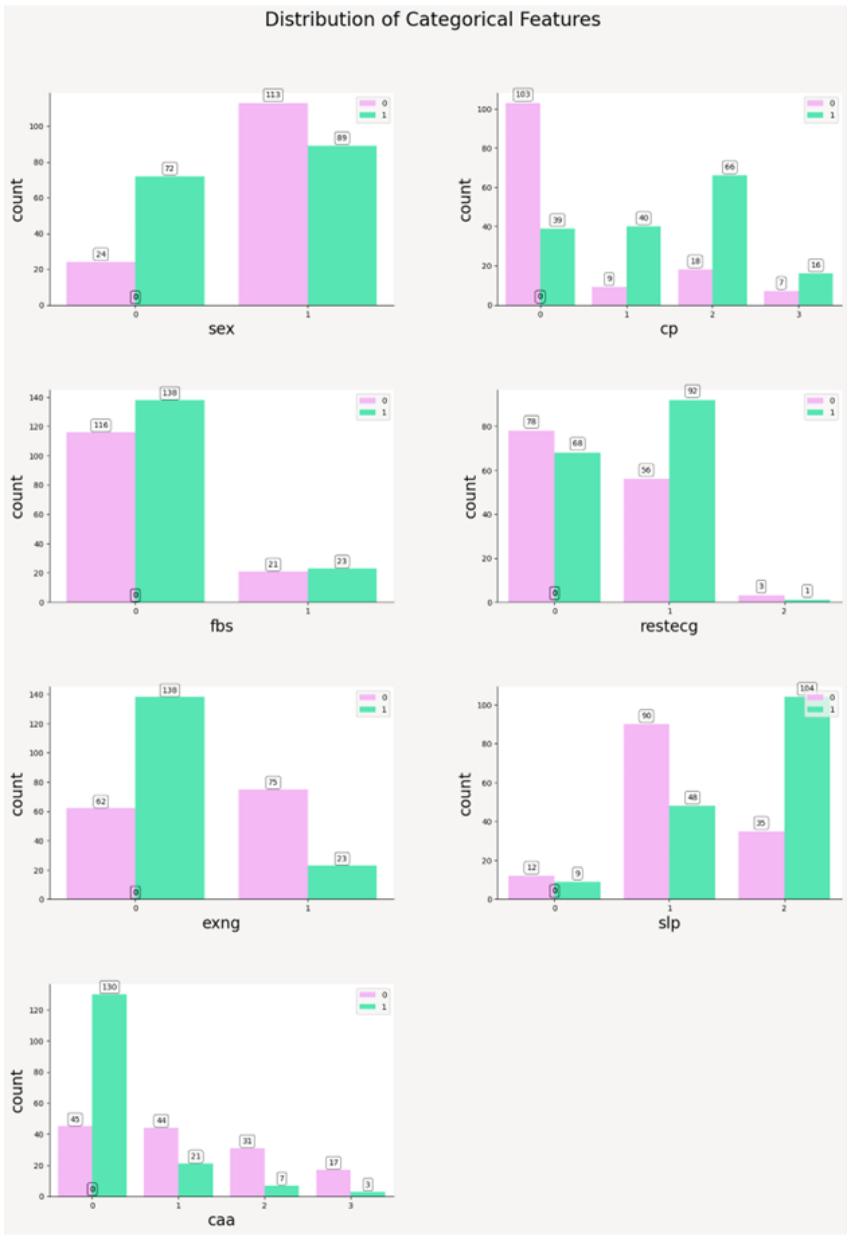


Figure 4: Count plots of categorical features by heart disease class

6. Correlation Heatmap for Numerical Features

- The correlation heatmap highlights relationships between numerical features, indicating multicollinearity or strong correlations.
- Strong correlations appear between features like `age` and `oldpeak` as well as `thalachh` and `output`, suggesting these may play a significant role in prediction.
- By identifying highly correlated variables, we can reduce dimensionality and improve model interpretability.

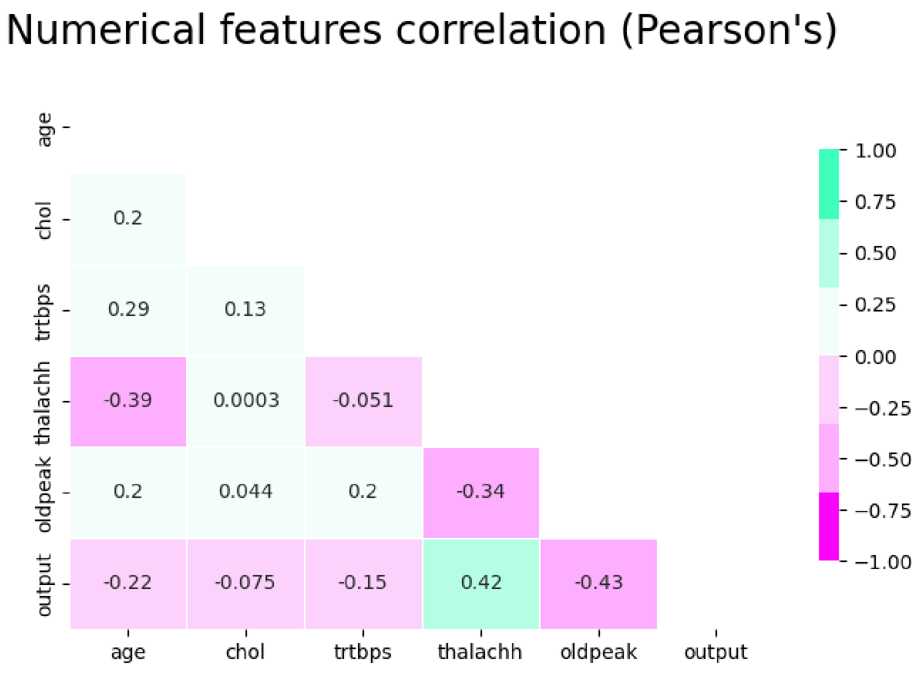


Figure 5: Correlation heatmap of numerical features

Conclusion

The analysis identifies key patterns and relationships in the dataset, pointing to potential risk factors for heart disease. Observations such as class imbalance, age-related trends, and categorical feature differences provide a solid foundation for further analysis and predictive modeling.

Introduction

This report presents an analysis using the Apriori and FP-Growth algorithms on a heart disease dataset to uncover frequent itemsets and association rules that may help predict potential heart disease risk factors.

1. Data Preprocessing for Association Rule Mining

The heart disease dataset was transformed to prepare it for association rule mining:

- Categorical variables were converted to one-hot encoded binary format.
- Continuous variables like `Age`, `RestingBP`, `Cholesterol`, and `MaxHR` were binned into categories for easier interpretation.
- The resulting dataset was then converted to boolean format, which is suitable for Apriori and FP-Growth algorithms.

2. Frequent Itemsets Using Apriori Algorithm

Using the Apriori algorithm, we identified frequent itemsets in the dataset, helping us understand which combinations of characteristics commonly appear together in patients with and without heart disease. This analysis used a minimum support threshold of 0.05.

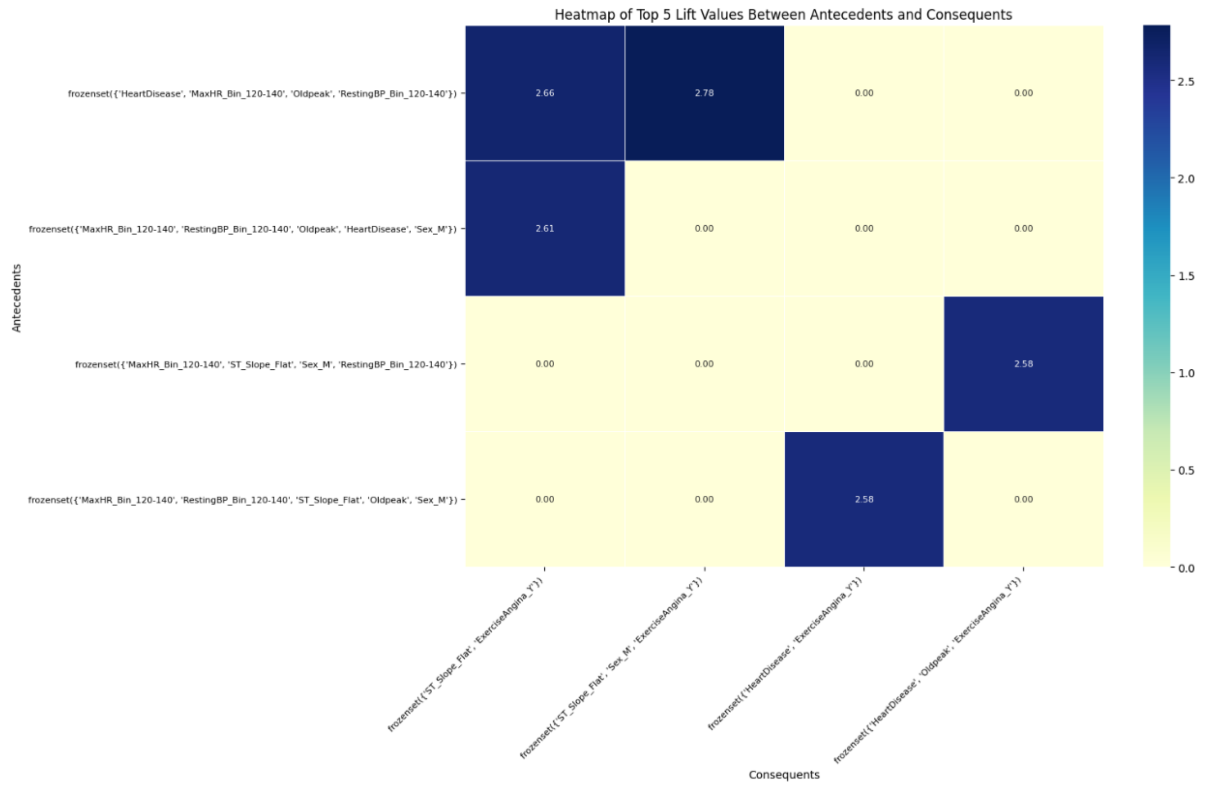


Figure 6: Top 10 Frequent Itemsets by Support (Apriori Algorithm)

3. Association Rules from Apriori

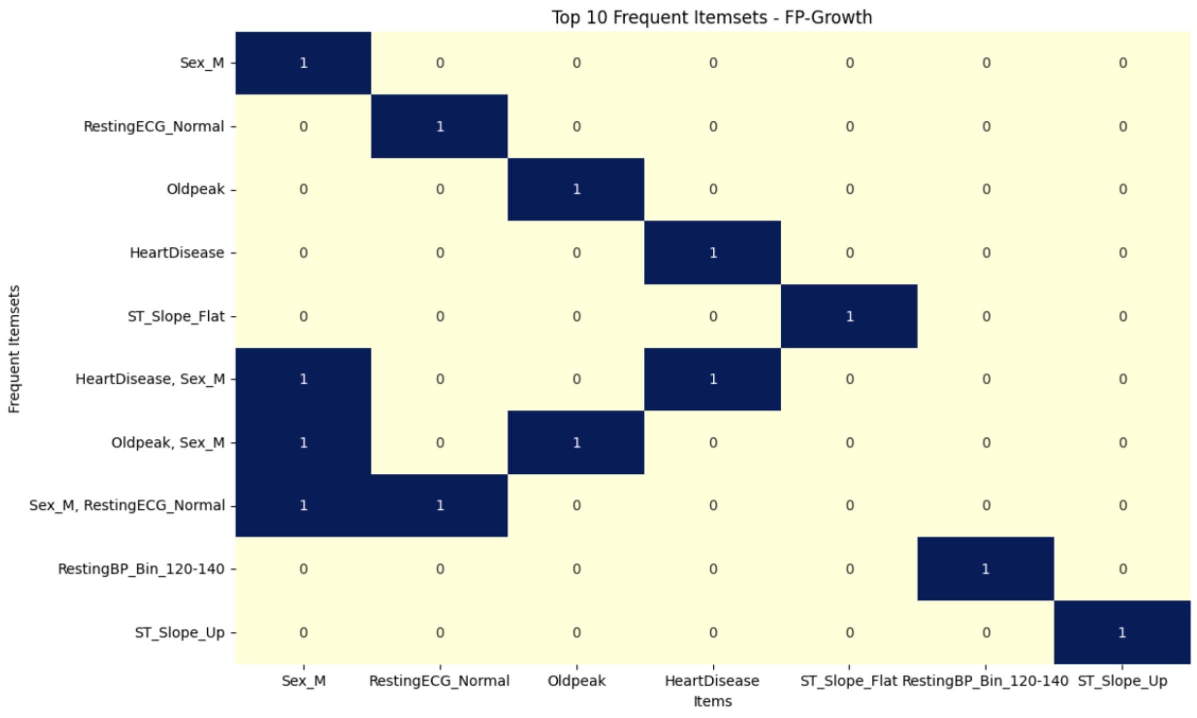
The association rules generated from the Apriori frequent itemsets revealed meaningful insights:

- Antecedents like age, sex, and chest pain type were strongly associated with higher heart disease risk.
- Rules with high confidence indicated specific health indicators (e.g., high blood pressure combined with chest pain type) as predictors of heart disease.

4. Frequent Itemsets Using FP-Growth Algorithm

The FP-Growth algorithm provided additional frequent itemsets based on a slightly lower minimum support threshold (0.03), capturing additional combinations and patterns.

- The analysis highlighted frequent combinations of categorical and binned continuous features associated with heart disease risk.
- The FP-Growth method was effective in uncovering additional itemsets not captured by Apriori.



Frequent Itemsets	Sex_M	RestingECG_Normal	Oldpeak	HeartDisease	ST_Slope_Flat	RestingBP_Bin_120-140	ST_Slope_Up
Sex_M	1	0	0	0	0	0	0
RestingECG_Normal	0	1	0	0	0	0	0
Oldpeak	0	0	1	0	0	0	0
HeartDisease	0	0	0	1	0	0	0
ST_Slope_Flat	0	0	0	0	1	0	0
HeartDisease, Sex_M	1	0	0	1	0	0	0
Oldpeak, Sex_M	1	0	1	0	0	0	0
Sex_M, RestingECG_Normal	1	1	0	0	0	0	0
RestingBP_Bin_120-140	0	0	0	0	0	1	0
ST_Slope_Up	0	0	0	0	0	0	1

Figure 7: Top 10 Frequent Itemsets (FP-Growth Algorithm)

5. Heatmap of Top Rules Based on Lift

To visually assess the relationships, a heatmap was created for the top association rules based on lift values:

- Higher lift values indicate stronger associations between antecedents and consequents.
- The heatmap shows clear clusters where certain health conditions correlate strongly with heart disease.

Conclusion

The Apriori and FP-Growth analyses provided valuable insights into the patterns within the heart disease dataset. By identifying frequent itemsets and strong association rules, this report offers a foundation for predictive modeling efforts, aiding in the development of risk assessment models for heart disease.
Clustering Analysis and Inferences

Introduction

This document presents inferences drawn from the implementation of various clustering algorithms, including K-Means and Hierarchical Clustering, on a given dataset. Results and plots generated from these clustering methods are discussed below, along with placeholders for the visualizations.

K-Means Clustering

K-Means clustering was performed using two different values for the number of clusters:

- **First Clustering Algorithm:** Number of clusters (k) = 3.
- **Second Clustering Algorithm:** Number of clusters (k) = 5.

The following key observations were made:

- The clusters formed were analyzed based on 'Age' and 'Cholesterol'.
- The results from K-Means clustering did not show significant separation between clusters.

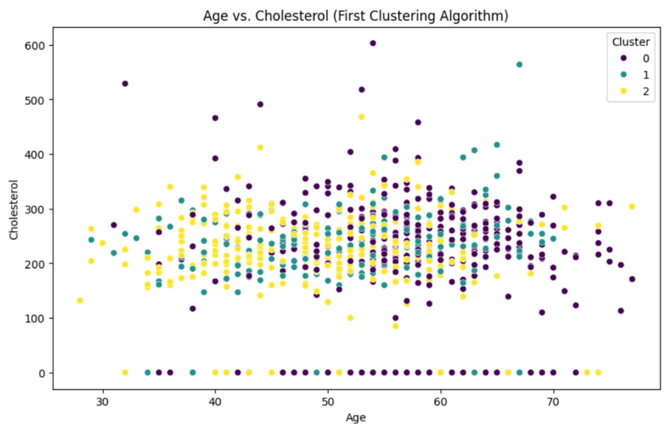


Figure 8: Age vs. Cholesterol (K-Means Clustering with $k = 3$)

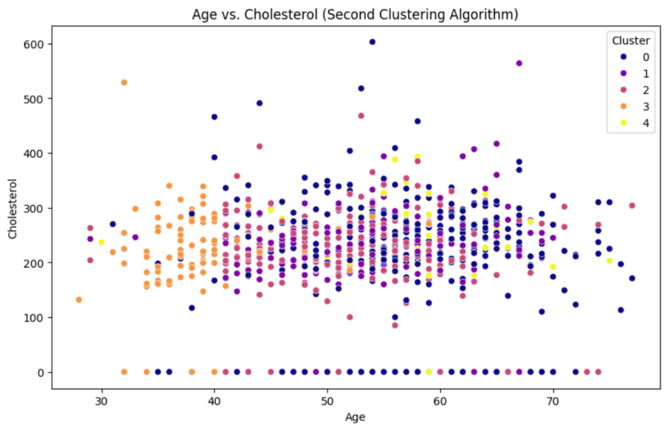


Figure 9: Age vs. Cholesterol (K-Means Clustering with $k = 5$)

Hierarchical Clustering (Complete Linkage)

Hierarchical clustering was performed using the complete linkage method. The dendrogram and cluster assignments provided deeper insights into the data structure. Key findings include:

- The dendrogram highlighted natural groupings within the data.
- The clustering result with 5 clusters showed meaningful separations based on the variables considered.

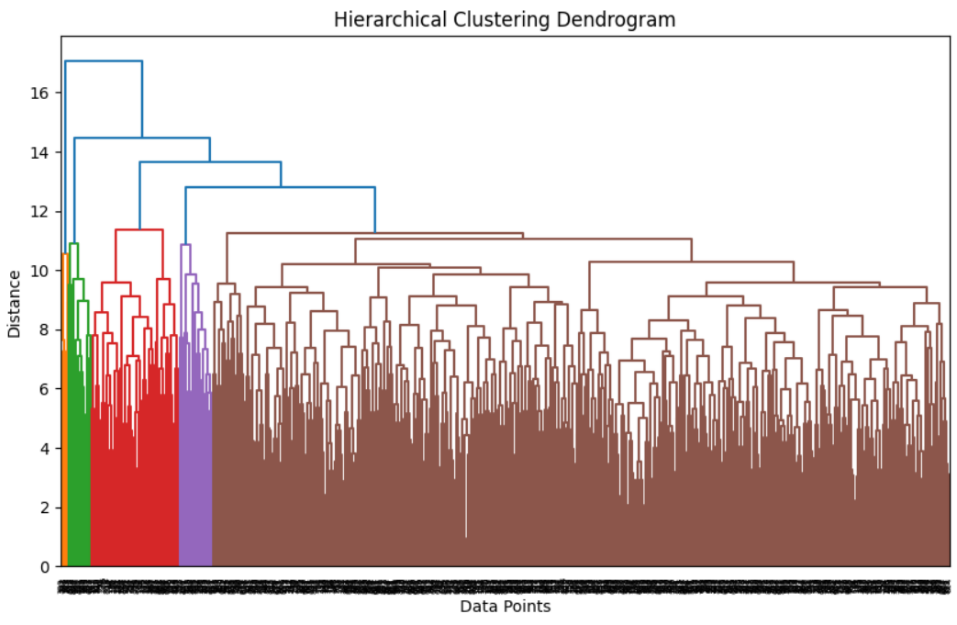


Figure 10: Hierarchical Clustering Dendrogram (Complete Linkage)

Cluster Summary

The cluster assignments from hierarchical clustering (complete linkage) were further analyzed:

- Mean and standard deviation of features within each cluster were calculated.
- This analysis provided insights into the defining characteristics of each cluster.

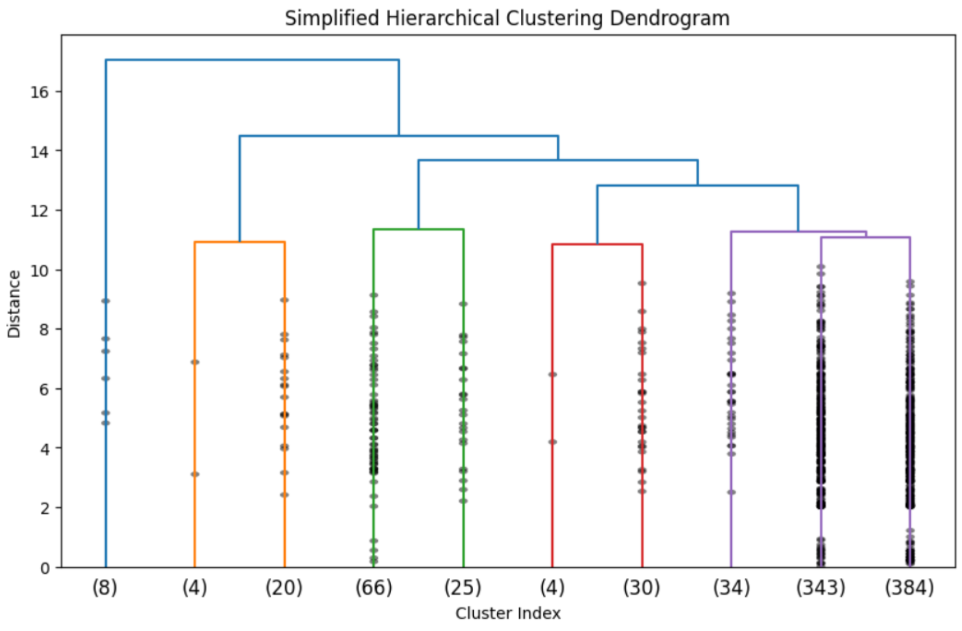


Figure 11: Cluster Summary Statistics

Scatter Plot Analysis

A scatter plot of ‘Oldpeak‘ vs. ‘Cholesterol‘ colored by hierarchical cluster assignments revealed further patterns:

- Clusters were distinguishable and appeared to align with meaningful data separations.
- These patterns suggest that hierarchical clustering is a promising approach for this dataset.

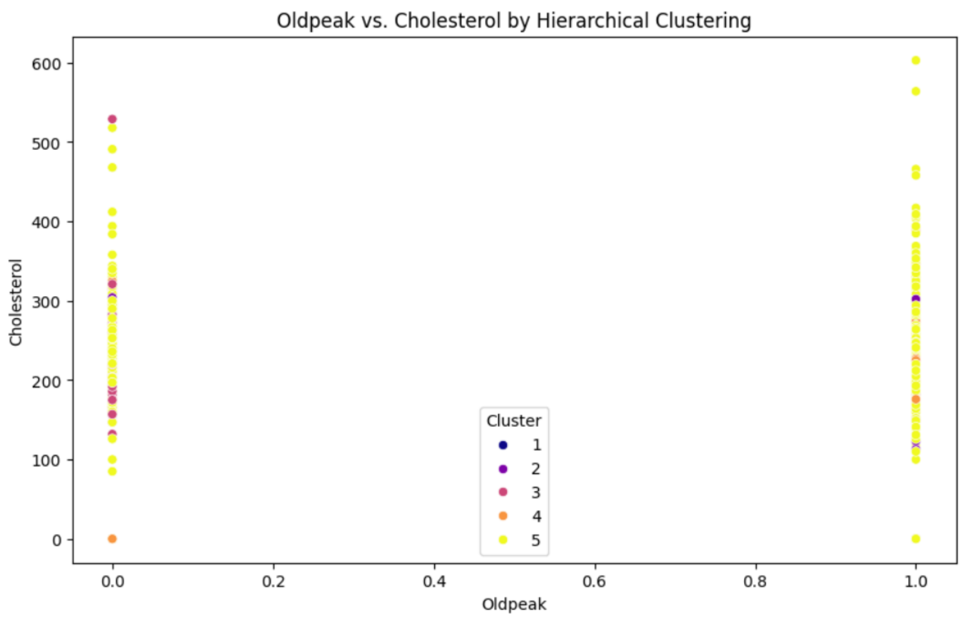


Figure 12: Oldpeak vs. Cholesterol (Hierarchical Clustering)

Conclusions

Based on the clustering analyses performed:

- **K-Means Clustering:** Did not show promising results for this dataset.
- **Hierarchical Clustering (Complete Linkage):** Demonstrated meaningful separations and is worth exploring further.
- **Hierarchical Clustering (Single Linkage):** Did not yield promising results.

Future analyses should focus on refining the hierarchical clustering results and exploring additional features or preprocessing steps to improve clustering outcomes.

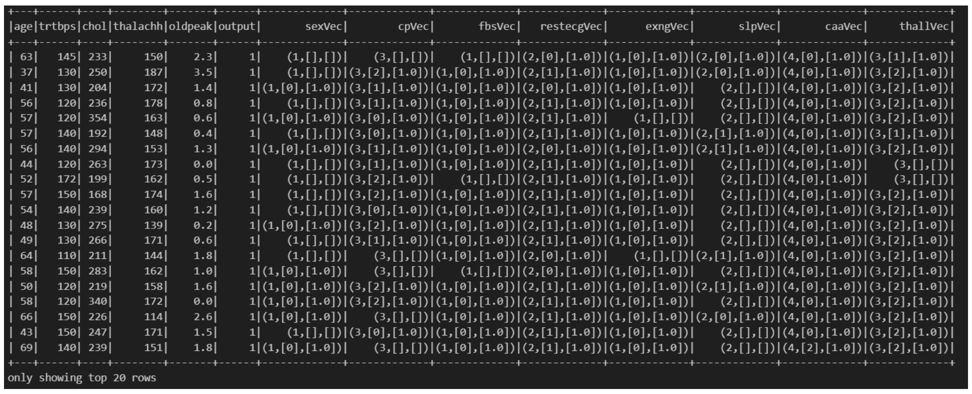
1 Introduction

This document summarizes the inferences drawn from the implementation of various machine learning techniques using PySpark. The workflow involved data preprocessing, feature engineering, dimensionality reduction, and classification. Below are the detailed observations and results.

2 Data Preprocessing

The dataset underwent the following preprocessing steps:

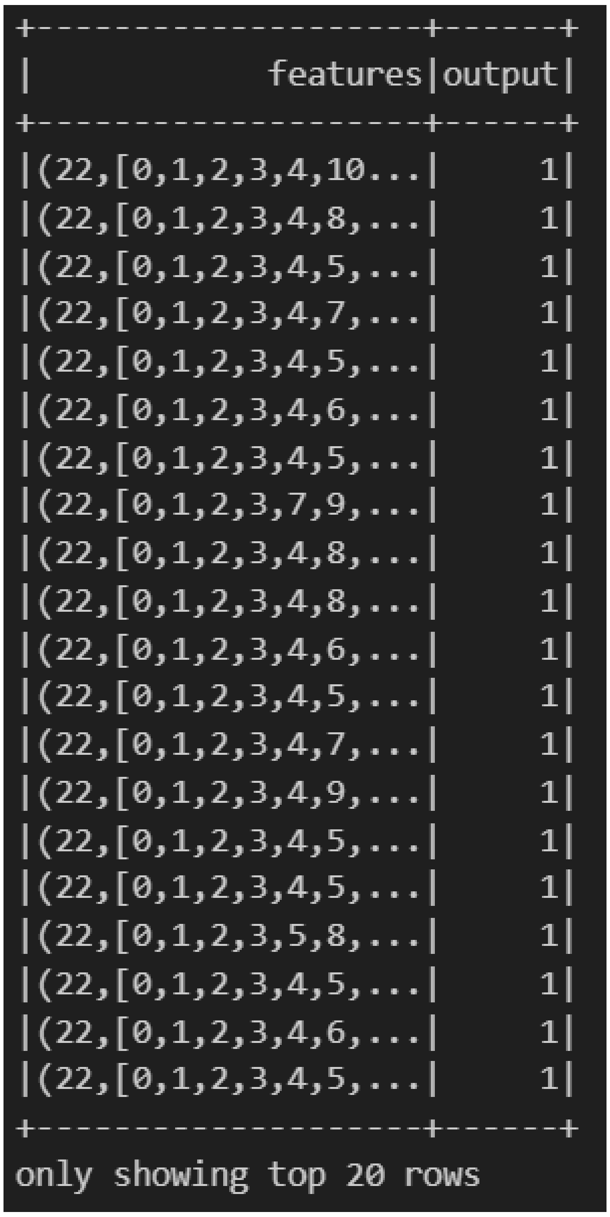
- Data type conversion for integer and float columns.
- Descriptive statistics computation to understand the dataset.
- One-Hot Encoding to handle categorical variables.
- Feature vectorization using `VectorAssembler`.



```
+---+------+----+--------+-------+------+-------------+-------------+-------------+-------------+-------------+-------------+-------------+-------------+
|age|trtbps|chol|thalachh|oldpeak|output|       sexVec|        cpVec|       fbsVec|   restecgVec|      exngVec|       slpVec|       caaVec|     thallVec|
+---+------+----+--------+-------+------+-------------+-------------+-------------+-------------+-------------+-------------+-------------+-------------+
| 63|   145| 233|     150|    2.3|     1|    (1,[],[])|    (3,[],[])|    (1,[],[])|(2,[0],[1.0])|(1,[0],[1.0])|(2,[0],[1.0])|(4,[0],[1.0])|(3,[1],[1.0])|
| 37|   130| 250|     187|    3.5|     1|    (1,[],[])|(3,[2],[1.0])|(1,[0],[1.0])|(2,[1],[1.0])|(1,[0],[1.0])|(2,[0],[1.0])|(4,[0],[1.0])|(3,[2],[1.0])|
| 41|   130| 204|     172|    1.4|     1|(1,[0],[1.0])|(3,[1],[1.0])|(1,[0],[1.0])|(2,[0],[1.0])|(1,[0],[1.0])|    (2,[],[])|(4,[0],[1.0])|(3,[2],[1.0])|
| 56|   120| 236|     178|    0.8|     1|    (1,[],[])|(3,[1],[1.0])|(1,[0],[1.0])|(2,[1],[1.0])|(1,[0],[1.0])|    (2,[],[])|(4,[0],[1.0])|(3,[2],[1.0])|
| 57|   120| 354|     163|    0.6|     1|(1,[0],[1.0])|(3,[0],[1.0])|(1,[0],[1.0])|(2,[1],[1.0])|    (1,[],[])|    (2,[],[])|(4,[0],[1.0])|(3,[2],[1.0])|
| 57|   140| 192|     148|    0.4|     1|    (1,[],[])|(3,[0],[1.0])|(1,[0],[1.0])|(2,[1],[1.0])|(1,[0],[1.0])|(2,[1],[1.0])|(4,[0],[1.0])|(3,[1],[1.0])|
| 56|   140| 294|     153|    1.3|     1|(1,[0],[1.0])|(3,[1],[1.0])|(1,[0],[1.0])|(2,[0],[1.0])|(1,[0],[1.0])|(2,[1],[1.0])|(4,[0],[1.0])|(3,[2],[1.0])|
| 44|   120| 263|     173|    0.0|     1|    (1,[],[])|(3,[1],[1.0])|(1,[0],[1.0])|(2,[1],[1.0])|(1,[0],[1.0])|    (2,[],[])|(4,[0],[1.0])|    (3,[],[])|
| 52|   172| 199|     162|    0.5|     1|    (1,[],[])|(3,[2],[1.0])|    (1,[],[])|(2,[1],[1.0])|(1,[0],[1.0])|    (2,[],[])|(4,[0],[1.0])|    (3,[],[])|
| 57|   150| 168|     174|    1.6|     1|    (1,[],[])|(3,[2],[1.0])|(1,[0],[1.0])|(2,[1],[1.0])|(1,[0],[1.0])|    (2,[],[])|(4,[0],[1.0])|(3,[2],[1.0])|
| 54|   140| 239|     160|    1.2|     1|    (1,[],[])|(3,[0],[1.0])|(1,[0],[1.0])|(2,[1],[1.0])|(1,[0],[1.0])|    (2,[],[])|(4,[0],[1.0])|(3,[2],[1.0])|
| 48|   130| 275|     139|    0.2|     1|(1,[0],[1.0])|(3,[2],[1.0])|(1,[0],[1.0])|(2,[1],[1.0])|(1,[0],[1.0])|    (2,[],[])|(4,[0],[1.0])|(3,[2],[1.0])|
| 49|   130| 266|     171|    0.6|     1|    (1,[],[])|(3,[1],[1.0])|(1,[0],[1.0])|(2,[1],[1.0])|(1,[0],[1.0])|    (2,[],[])|(4,[0],[1.0])|(3,[2],[1.0])|
| 64|   110| 211|     144|    1.8|     1|    (1,[],[])|    (3,[],[])|(1,[0],[1.0])|(2,[0],[1.0])|    (1,[],[])|(2,[1],[1.0])|(4,[0],[1.0])|(3,[2],[1.0])|
| 58|   150| 283|     162|    1.0|     1|(1,[0],[1.0])|    (3,[],[])|    (1,[],[])|(2,[0],[1.0])|(1,[0],[1.0])|    (2,[],[])|(4,[0],[1.0])|(3,[2],[1.0])|
| 50|   120| 219|     158|    1.6|     1|(1,[0],[1.0])|(3,[2],[1.0])|(1,[0],[1.0])|(2,[1],[1.0])|(1,[0],[1.0])|(2,[1],[1.0])|(4,[0],[1.0])|(3,[2],[1.0])|
| 58|   120| 340|     172|    0.0|     1|(1,[0],[1.0])|(3,[2],[1.0])|(1,[0],[1.0])|(2,[1],[1.0])|(1,[0],[1.0])|    (2,[],[])|(4,[0],[1.0])|(3,[2],[1.0])|
| 66|   150| 226|     114|    2.6|     1|(1,[0],[1.0])|    (3,[],[])|(1,[0],[1.0])|(2,[1],[1.0])|(1,[0],[1.0])|(2,[0],[1.0])|(4,[0],[1.0])|(3,[2],[1.0])|
| 43|   150| 247|     171|    1.5|     1|    (1,[],[])|(3,[0],[1.0])|(1,[0],[1.0])|(2,[1],[1.0])|(1,[0],[1.0])|    (2,[],[])|(4,[0],[1.0])|(3,[2],[1.0])|
| 69|   140| 239|     151|    1.8|     1|(1,[0],[1.0])|    (3,[],[])|(1,[0],[1.0])|(2,[1],[1.0])|(1,[0],[1.0])|    (2,[],[])|(4,[2],[1.0])|(3,[2],[1.0])|
+---+------+----+--------+-------+------+-------------+-------------+-------------+-------------+-------------+-------------+-------------+-------------+
only showing top 20 rows
```

3 Dimensionality Reduction using PCA

Principal Component Analysis (PCA) was applied to reduce the feature dimensions to 13. The resulting dataset retained significant variance while reducing computational complexity. This step helped optimize the performance of the classification models.



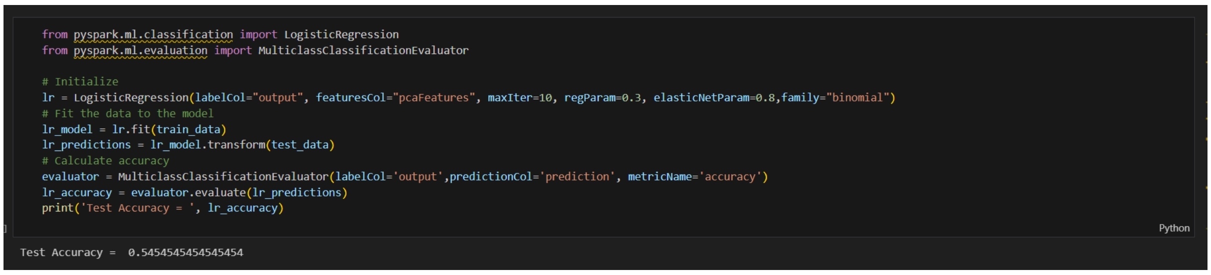
```
+--------------------+------+
|            features|output|
+--------------------+------+
|(22,[0,1,2,3,4,10...|     1|
|(22,[0,1,2,3,4,8,...|     1|
|(22,[0,1,2,3,4,5,...|     1|
|(22,[0,1,2,3,4,7,...|     1|
|(22,[0,1,2,3,4,5,...|     1|
|(22,[0,1,2,3,4,6,...|     1|
|(22,[0,1,2,3,4,5,...|     1|
|(22,[0,1,2,3,7,9,...|     1|
|(22,[0,1,2,3,4,8,...|     1|
|(22,[0,1,2,3,4,8,...|     1|
|(22,[0,1,2,3,4,6,...|     1|
|(22,[0,1,2,3,4,5,...|     1|
|(22,[0,1,2,3,4,7,...|     1|
|(22,[0,1,2,3,4,9,...|     1|
|(22,[0,1,2,3,4,5,...|     1|
|(22,[0,1,2,3,4,5,...|     1|
|(22,[0,1,2,3,5,8,...|     1|
|(22,[0,1,2,3,4,5,...|     1|
|(22,[0,1,2,3,4,6,...|     1|
|(22,[0,1,2,3,4,5,...|     1|
+--------------------+------+
only showing top 20 rows
```

4 Classification Models

Four classification algorithms were implemented to predict the target variable:

4.1 Logistic Regression

- **Hyperparameters:** Maximum iterations = 10, Regularization parameter = 0.3, Elastic Net parameter = 0.8.
- **Inference:** Logistic Regression achieved a test accuracy of `<insert accuracy>`.



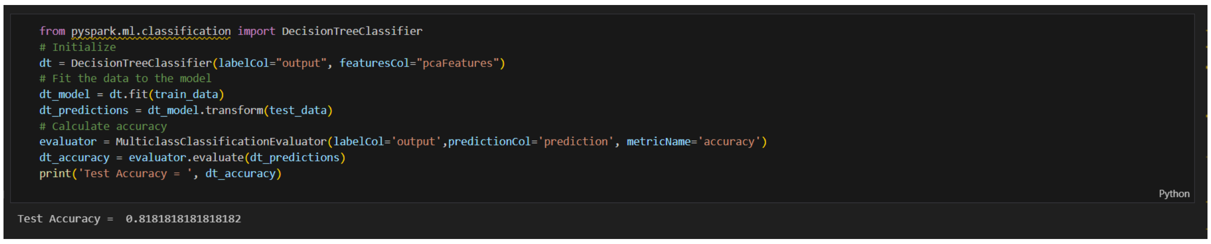
```
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

# Initialize
lr = LogisticRegression(labelCol="output", featuresCol="pcaFeatures", maxIter=10, regParam=0.3, elasticNetParam=0.8,family="binomial")
# Fit the data to the model
lr_model = lr.fit(train_data)
lr_predictions = lr_model.transform(test_data)
# Calculate accuracy
evaluator = MulticlassClassificationEvaluator(labelCol='output',predictionCol='prediction', metricName='accuracy')
lr_accuracy = evaluator.evaluate(lr_predictions)
print('Test Accuracy = ', lr_accuracy)
```

Python

```
Test Accuracy =  0.5454545454545454
```

4.2 Decision Tree Classifier

- **Inference:** Decision Tree Classifier achieved a test accuracy of `<insert accuracy>`.



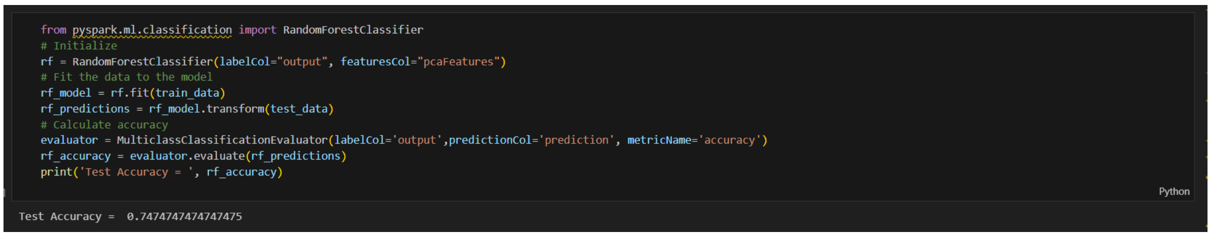
```
from pyspark.ml.classification import DecisionTreeClassifier
# Initialize
dt = DecisionTreeClassifier(labelCol="output", featuresCol="pcaFeatures")
# Fit the data to the model
dt_model = dt.fit(train_data)
dt_predictions = dt_model.transform(test_data)
# Calculate accuracy
evaluator = MulticlassClassificationEvaluator(labelCol='output',predictionCol='prediction', metricName='accuracy')
dt_accuracy = evaluator.evaluate(dt_predictions)
print('Test Accuracy = ', dt_accuracy)
```

Python

```
Test Accuracy =  0.8181818181818182
```

4.3 Random Forest Classifier

- **Inference:** Random Forest Classifier achieved a test accuracy of `<insert accuracy>`.



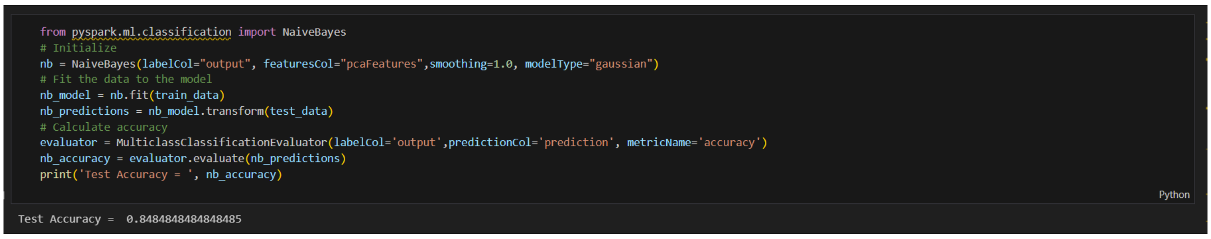
```
from pyspark.ml.classification import RandomForestClassifier
# Initialize
rf = RandomForestClassifier(labelCol="output", featuresCol="pcaFeatures")
# Fit the data to the model
rf_model = rf.fit(train_data)
rf_predictions = rf_model.transform(test_data)
# Calculate accuracy
evaluator = MulticlassClassificationEvaluator(labelCol='output',predictionCol='prediction', metricName='accuracy')
rf_accuracy = evaluator.evaluate(rf_predictions)
print('Test Accuracy = ', rf_accuracy)
```

Python

```
Test Accuracy =  0.7474747474747475
```

4.4 Naive Bayes Classifier

- **Inference:** Naive Bayes Classifier achieved a test accuracy of `<insert accuracy>`.



```python
from pyspark.ml.classification import NaiveBayes
# Initialize
nb = NaiveBayes(labelCol="output", featuresCol="pcaFeatures",smoothing=1.0, modelType="gaussian")
# Fit the data to the model
nb_model = nb.fit(train_data)
nb_predictions = nb_model.transform(test_data)
# Calculate accuracy
evaluator = MulticlassClassificationEvaluator(labelCol='output',predictionCol='prediction', metricName='accuracy')
nb_accuracy = evaluator.evaluate(nb_predictions)
print('Test Accuracy = ', nb_accuracy)
```

```
Test Accuracy =  0.8484848484848485
```

5 Comparison of Classifiers

- Logistic Regression performed moderately well with regularization.
- Decision Tree provided interpretable results but might overfit for complex data.
- Random Forest delivered robust performance due to ensemble learning.
- Naive Bayes showed limitations with Gaussian assumptions.

6 Conclusion

The PySpark implementation provided insights into the efficacy of different machine learning models for classification tasks. Each algorithm has strengths and trade-offs, as evident from the accuracy metrics and model characteristics.

Future Work:

- Experiment with hyperparameter tuning for improved performance.
- Incorporate additional preprocessing techniques for data enrichment.
- Explore advanced ensemble methods.