Project Big Data Processing

Orsino Jerrell Bruinier - 2602065540
Talethia Hayfa Syukur - 2602067016
Jonathan Theja - 2602055634
Maura Andini Sunusmo - 2602065396
Angel Priscilla Salim - 2602055281

Judul Project

Prediksi Klasifikasi Stadium Sirosis Hati menggunakan Spark Decision Tree dan Random Forest

Daftar Isi

Latar Belakang

Sirosis hati merupakan penyakit hati dengan kondisi yang sangat serius di mana ditandai dengan kerusakan hati yang digantikan oleh jaringan parut secara permanen.

Sirosis hati merupakan peringkat 5 penyakit yang menjadi penyebab kematian tertinggi di Indonesia. Hepatitis B dan C adalah penyakit infeksi yang masih menjadi penyebab utama sirosis hati di Indonesia.

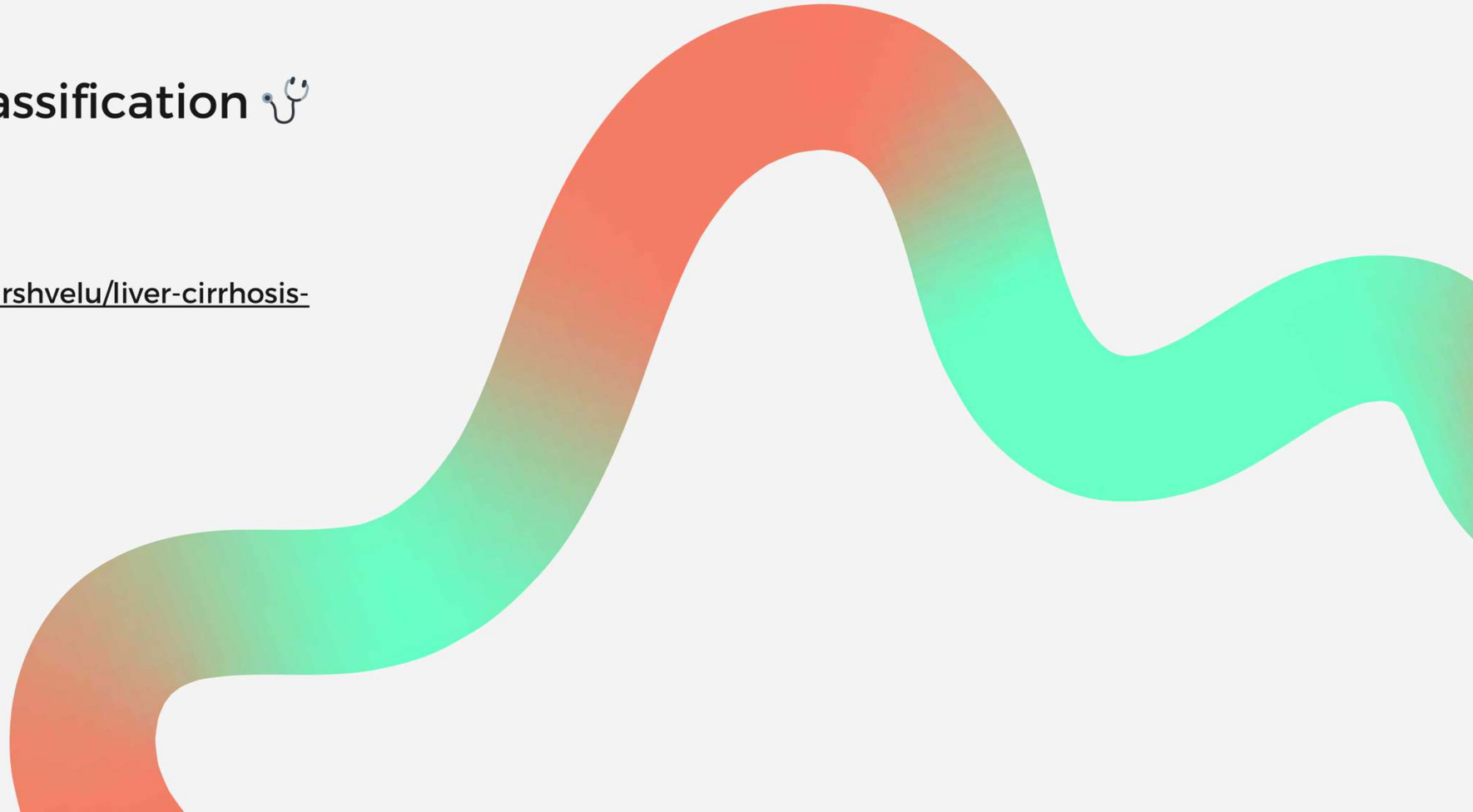
Tiga stadium atau tahap dalam penyakit sitosis hati, yaitu stadium satu(kompensasi), stadium dua(dekompensasi), dan stadium tiga(decompenata).

Dataset

Liver Cirrhosis Stage Classification 🩺

By : Aadarsh Velu

https://www.kaggle.com/datasets/aadarshvelu/liver-cirrhosis-stage-classification



Metodologi dan Alur Pekerjaan

Analytic Flow Big Data

Data collected dari Kaggle dataset melalui Kaggle API.

Berbagai preprocessing dilakukan dan data splitting menjadi 80% training dan 20% testing

Analisa Prediktif
Supervised Learning
Prediksi target variabel berlabel

Classification Model

Berbagai visualisasi digunakan untuk menampilkan performance

Metodologi dan Alur Pekerjaan

1 Data Preparation

```
[3] # Konfigurasi Kaggle API
    os.environ['KAGGLE_CONFIG_DIR'] = os.path.expanduser('~/.kaggle')

    # Mengunduh dataset dengan Kaggle API
    dataset_identifier = 'aadarshvelu/liver-cirrhosis-stage-classification'
    subprocess.run(['kaggle', 'datasets', 'download', '-d', dataset_identifier, '-p', './'])

    # Unzip dataset
    import zipfile
    with zipfile.ZipFile('./liver-cirrhosis-stage-classification.zip', 'r') as zip_ref:
        zip_ref.extractall('./')

    extracted_files = os.listdir('./')
    print("Extracted files:", extracted_files)

    csv_file = [file for file in extracted_files if file.endswith('.csv')][0]
    file_path = f'./{csv_file}'
    print("CSV File Path:", file_path)
```

```
Extracted files: ['.config', 'liver-cirrhosis-stage-classification.zip', 'liver_cirrhosis.csv', 'sample_data']
CSV File Path: ./liver_cirrhosis.csv
```

- Menetapkan direktori untuk konfigurasi API Kaggle
- Mengunduh dataset melalui API Kaggle
- ZIP yang ditemukan kemudian di ekstrak dan mencari file CSV dalam ZIP tersebut.
- Dataset sudah dapat diproses dan analisis

Metodologi dan Alur Pekerjaan

2 Creating Spark Session and Data Loading

```
[4]  # Membuat Spark session
     spark = SparkSession.builder \
         .appName("Liver Cirrhosis Stage Classification") \
         .getOrCreate()

     # Load dataset
     df = spark.read.csv(file_path, header=True, inferSchema=True)

     # Menampilkan schema
     df.printSchema()
     df.show(20)
```

- Membuat sesi spark dengan nama “Liver Cirrhosis Stage Classification”.
- Membuat dataset CSV ke dalam Data Frame Spark.
- Menampilkan skema dataset dan 20 baris pertama untuk verifikasi data.

Metodologi dan Alur Pekerjaan

3 Data Preprocessing

```
# Cek NULL values pada tabel
null_counts_dict = {col_name: df.filter(col(col_name).isNull()).count() for col_name in df.columns}

null_counts_df = pd.DataFrame(list(null_counts_dict.items()), columns=['Column', 'Null Values'])

# Display DataFrame
print(null_counts_df)
```

```
           Column  Null Values
0          N_Days            0
1          Status            0
2            Drug            0
3             Age            0
4             Sex            0
5         Ascites            0
6    Hepatomegaly            0
7         Spiders            0
8           Edema            0
9       Bilirubin            0
10    Cholesterol            0
11        Albumin            0
12         Copper            0
13       Alk_Phos            0
14           SGOT            0
15  Tryglicerides            0
16      Platelets            0
17    Prothrombin            0
18          Stage            0
```

- Menampilkan NULL Value pada dataset

Metodologi dan Alur Pekerjaan

3 Data Preprocessing

```
# Data preprocessing

# Periksa Nilai Unik pada 'Stage'
df_cleaned.groupBy("Stage").count().show()

# Ubah umur dari hari ke tahun
df_cleaned = df_cleaned.withColumn("Age_Years", df_cleaned["Age"] / 365)

# Identifikasi Kolom Categorical
categorical_columns = [field for (field, dataType) in df_cleaned.dtypes if dataType == "string"]

# Terapkan StringIndexer pada semua kolom Categorical
indexers = [StringIndexer(inputCol=column, outputCol=column+"_indexed") for column in categorical_columns]
pipeline = Pipeline(stages=indexers)
df_indexed = pipeline.fit(df_cleaned).transform(df_cleaned)

# Perbarui kolom feature untuk menyertakan kolom Categorical yang di Index
feature_columns = [column for column in df.columns if column != "Stage" and column not in categorical_columns and column != "Age"]
feature_columns.append("Age_Years")
feature_columns += [column+"_indexed" for column in categorical_columns]

# Membuat features menjadi satu vector
assembler = VectorAssembler(inputCols=feature_columns, outputCol="unscaled_features")
df_assembled = assembler.transform(df_indexed).select("unscaled_features", "Stage")

# Scale Fitur
scaler = StandardScaler(inputCol="unscaled_features", outputCol="features")
df_final = scaler.fit(df_assembled).transform(df_assembled).select("features", "Stage")

# Data Splitting
train_data, test_data = df_final.randomSplit([0.8, 0.2], seed=1234)
```

```
+-----+-----+
|Stage|count|
+-----+-----+
|    1| 8265|
|    3| 8294|
|    2| 8441|
+-----+-----+
```

- Konversi atribut Age yang tadinya dalam hari menjadi dalam tahun
- Konversi kolom kategorikal menjadi numerik menggunakan StringIndexer
- Menyatukan fitur - fitur menjadi satu vektor menggunakan VectorAssembler.
- Fitur diskalakan menggunakan StandardScaler untuk normalisasi data agar menghindari dominasi fitur dengan rentang nilai yang lebih besar.
- Data set dibagi menjadi 80% untuk training dan 20% untuk testing.

Metodologi dan Alur Pekerjaan

4 Model Generation

```
# Define model yang digunakan
dt = DecisionTreeClassifier()
rf = RandomForestClassifier()

# Define parameter grids untuk hyperparameter tuning
param_grid_dt = ParamGridBuilder() \
    .addGrid(dt.maxDepth, [5, 10, 15, 20]) \
    .build()

param_grid_rf = ParamGridBuilder() \
    .addGrid(rf.numTrees, [50, 100, 200]) \
    .addGrid(rf.maxDepth, [5, 10, 15, 20]) \
    .build()

# Define pipelines
dt_pipeline = Pipeline(stages=[dt])
rf_pipeline = Pipeline(stages=[rf])

# Define cross validators
crossval_dt = CrossValidator(estimator=dt_pipeline,
                             estimatorParamMaps=param_grid_dt,
                             evaluator=MulticlassClassificationEvaluator(metricName="accuracy"),
                             numFolds=3)

crossval_rf = CrossValidator(estimator=rf_pipeline,
                             estimatorParamMaps=param_grid_rf,
                             evaluator=MulticlassClassificationEvaluator(metricName="accuracy"),
                             numFolds=3)

# Model Training
dt_model = crossval_dt.fit(train_data)
rf_model = crossval_rf.fit(train_data)
```

- Mendefinisikan model Decision Tree dan Random Forest
- Menentukan Grid parameter untuk hyperparameter tuning menggunakan ‘PararmGridBuilder’.
- Membuat pipeline untuk Decision Tree dan Random Forest.
- Mendefinisikan Cross validator untuk kedua model dengan evaluator ‘MultiClassClassificationEvaluator’ berdasarkan akurasi.

Metodologi dan Alur Pekerjaan

5 Evaluation

```
# Buat Prediksi
dt_predictions = dt_model.transform(test_data)
rf_predictions = rf_model.transform(test_data)

# Evaluasi Model
evaluator = MulticlassClassificationEvaluator(metricName="accuracy")

dt_accuracy = evaluator.evaluate(dt_predictions)
rf_accuracy = evaluator.evaluate(rf_predictions)
```

- Membuat prediksi pada model yang telah dilatih menggunakan porsi test dari dataset.
- Menghitung akurasi model dengan menggunakan “MulticlassClassificationEvaluator”.

Metodologi dan Alur Pekerjaan

6 Visualization

```
# Function untuk plot confusion matrix
def plot_confusion_matrix(predictions, title):
    y_true = predictions.select("Stage").collect()  # Change 'label' to 'Stage'
    y_pred = predictions.select("prediction").collect()

    y_true = np.array([row[0] for row in y_true])
    y_pred = np.array([row[0] for row in y_pred])

    cm = confusion_matrix(y_true, y_pred)
    sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
    plt.xlabel('Predicted Label')
    plt.ylabel('True Label')
    plt.title(title)
    plt.show()

# Function untuk mengukur metrics :  precision, recall, and F1-score
def calculate_metrics(predictions):
    y_true = predictions.select("Stage").collect()  # Change 'label' to 'Stage'
    y_pred = predictions.select("prediction").collect()

    y_true = np.array([row[0] for row in y_true])
    y_pred = np.array([row[0] for row in y_pred])

    precision = precision_score(y_true, y_pred, average='weighted')
    recall = recall_score(y_true, y_pred, average='weighted')
    f1 = f1_score(y_true, y_pred, average='weighted')

    return precision, recall, f1

# Calculate metrics Decision Tree
dt_precision, dt_recall, dt_f1 = calculate_metrics(dt_predictions)
# Calculate metrics Random Forest
rf_precision, rf_recall, rf_f1 = calculate_metrics(rf_predictions)

# Print metrics Decision Tree
print(f"Decision Tree Accuracy: {dt_accuracy}")
print(f"Decision Tree Precision: {dt_precision}")
print(f"Decision Tree Recall: {dt_recall}")
print(f"Decision Tree F1-Score: {dt_f1}\n")

# Print metrics Random Forest
print(f"Random Forest Accuracy: {rf_accuracy}")
print(f"Random Forest Precision: {rf_precision}")
print(f"Random Forest Recall: {rf_recall}")
print(f"Random Forest F1-Score: {rf_f1}")

# Plot confusion matrices
plot_confusion_matrix(dt_predictions, "Decision Tree Confusion Matrix")
plot_confusion_matrix(rf_predictions, "Random Forest Confusion Matrix")
```

- Membuat Confusion Matrix dengan library seaborn.
- Menampilkan metrics seperti Precision, Recall, dan F1 - Score
- Membuat beberapa visualisasi lainya yang dilakukan untuk keperluan data analytics.

Evaluasi

Performance Metrics Table

No	Metrics	Model	
		Random Forest	Decision Tree
1	Accuracy	95.38985%	91.28081%
2	Precision	95.39406%	91.29407%
3	Recall	95.38985%	91.28081%
4	F1 - Score	95.38799%	91.2858%

- Accuracy : seberapa akurat suatu model dapat mengklasifikasikan dengan benar
- Precision : tingkat akurasi data yang diminta dengan hasil prediksi model yang digunakan
- Recall : akurasi hasil prediksi model yang dibandingkan dengan keseluruhan jumlah *ground truth*
- F1 - Score : perbandingan rata-rata harmonik precision dan recall

[Ground Truth]
Data yang dianggap benar secara objektif dan digunakan sebagai acuan untuk melatih, menguji, dan mengevaluasi model

Evaluasi

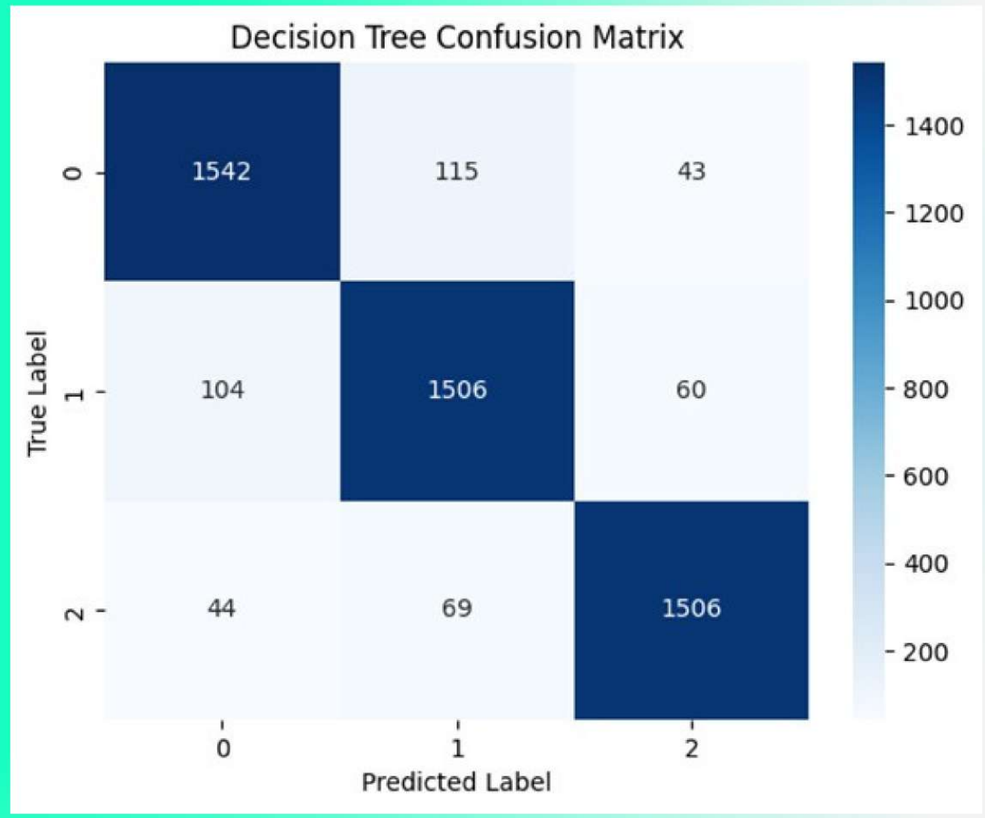
Random Forest VS Decision Tree

	Decision Tree	Random Forest
Overfitting	Cenderung mudah mengalami overfitting terhadap data training, sehingga performa model berkurang pada data baru	Diatasi dengan pendekatan ensemble (menggabungkan hasil dari banyak DT yang dibangun dengan random subset dari data)
Varians	Varians cenderung tinggi, sehingga perubahan kecil dalam training data dapat menghasilkan tree yang sangat berbeda	Diatasi dengan pendekatan ensemble untuk menghasilkan prediksi yang lebih stabil
Dimensi Data	Memerlukan tree yang sangat besar untuk menangkap kompleksitas data dan dapat menyebabkan overfitting	Diatasi dengan membangun banyak tree dan mempertimbangkan random subset dari fitur pada setiap split untuk menangkap struktur data tanpa menyebabkan overfitting
Noise dan Outliers	Sangat sensitive terhadap noise dan outliers	Ketahanan lebih baik, karena model merata-ratakan noise data pada banyak tree

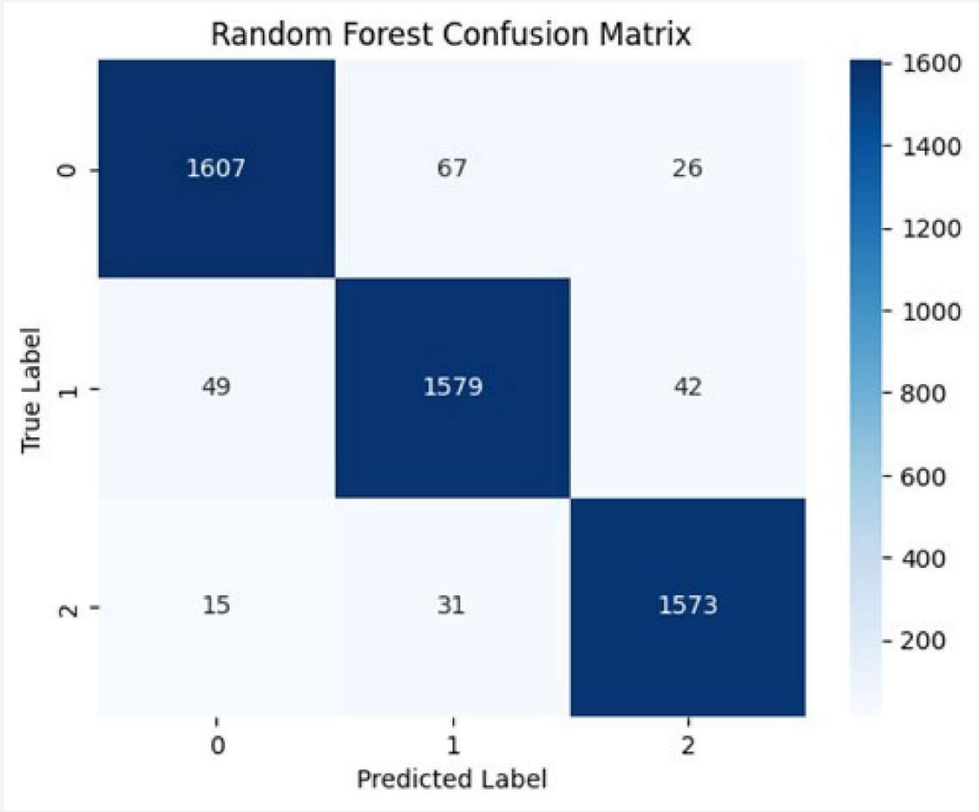
Evaluasi

Random Forest VS Decision Tree

4554 dari 4989 data (91.28%)

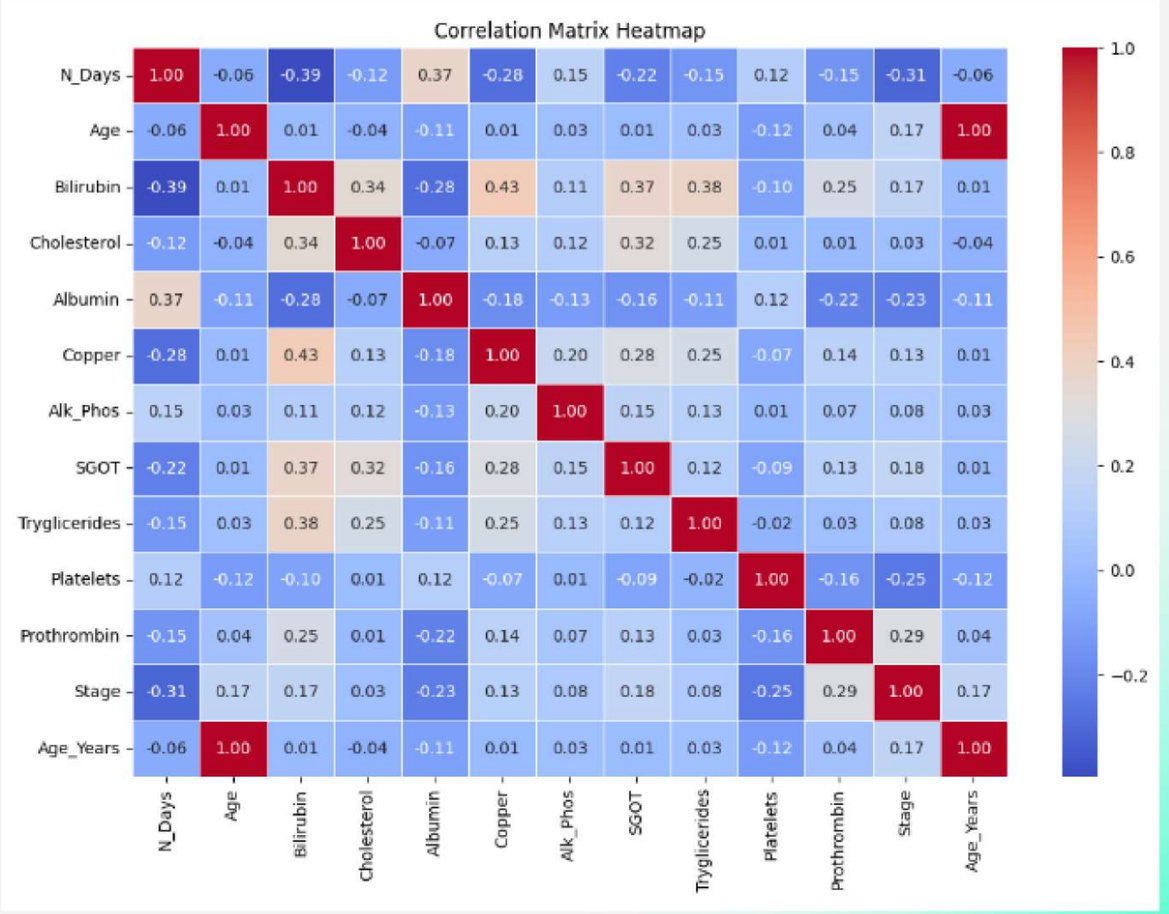


4759 dari 4989 data (95.39%)



Heatmap matriks korelasi

- **Korelasi Positif Kuat**: Seperti Bilirubin dan Copper (0.43), yang menunjukkan peningkatan bersamaan antara penanda biokimia ini, kemungkinan menunjukkan kenaikan yang terjadi bersamaan karena disfungsi hati.
- **Korelasi Moderat hingga Lemah**: Misalnya, N_Days dan Albumin (0.37) menunjukkan bahwa waktu bertahan hidup yang lebih lama umumnya dikaitkan dengan tingkat Albumin yang lebih tinggi, yang mungkin menandakan fungsi hati yang lebih baik dalam jangka waktu yang lebih lama.
- **Korelasi Negatif**: Menyoroti hubungan terbalik seperti antara Stage dan Albumin (-0.23) di mana tahap penyakit yang maju sesuai dengan penurunan kadar Albumin, menunjukkan penurunan fungsi hati.
- **Visualisasi yang Jelas**: Intensitas warna setiap sel menunjukkan kekuatan korelasi, memudahkan pengidentifikasian hubungan penting dengan cepat.



Bar Chart

1. **Asites (penumpukan cairan di perut)**

- Pada stadium 1 dan 2, kejadian asites lebih tinggi pada pasien tanpa asites (N) daripada yang memiliki asites (Y).
- Namun, pada stadium 3, jumlah pasien dengan asites (Y) lebih tinggi, menunjukkan peningkatan risiko asites pada stadium penyakit yang lebih lanjut.

2. **Hepatomegali (pembesaran hati)**

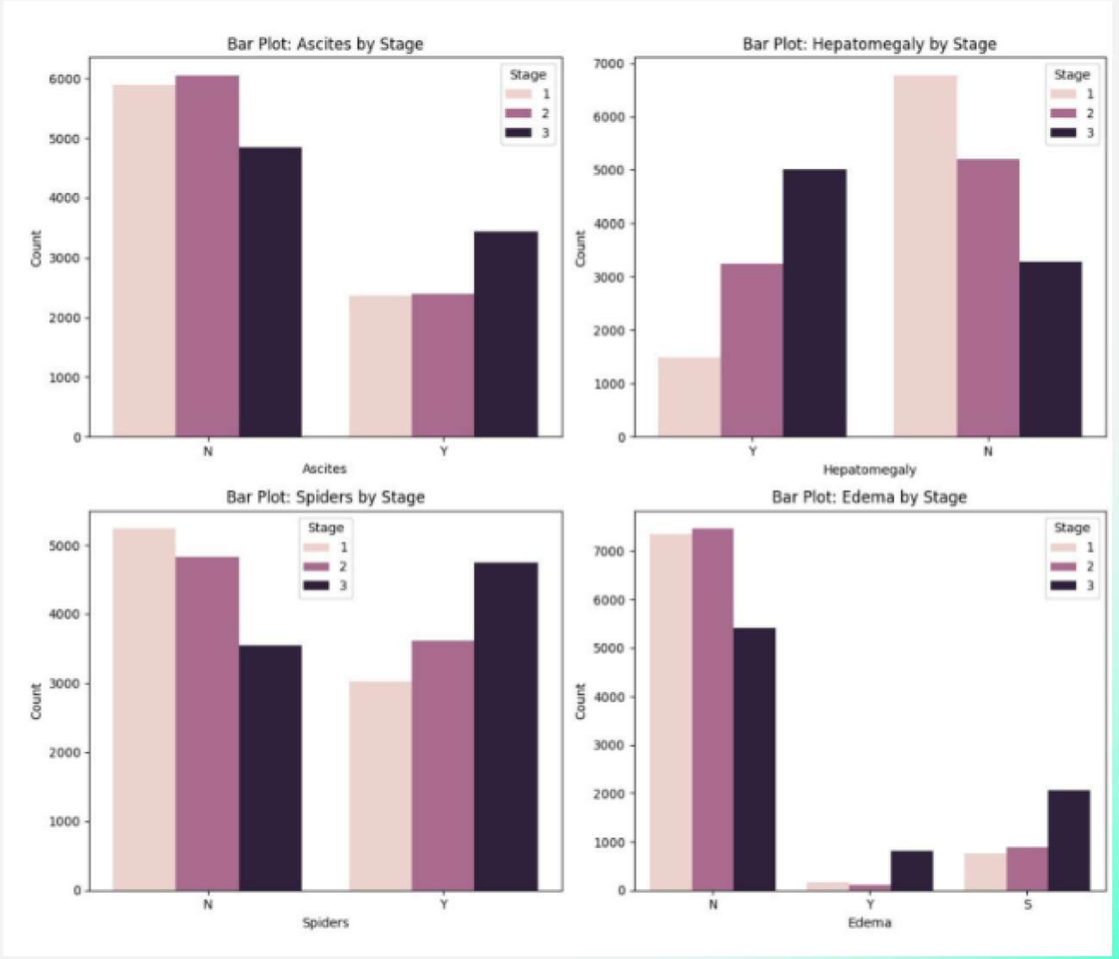
- Lebih sering terjadi pada stadium 3, diikuti oleh stadium 2, dan paling rendah pada stadium 1.
- Pasien tanpa hepatomegali (N) menunjukkan kecenderungan jumlah kasus menurun dari stadium 3 ke stadium 1.

3. **Spider nevi (pembuluh darah kecil melebar di kulit)**

- Pada pasien dengan spider nevi (Y), jumlah kasus meningkat dari stadium 1 ke stadium 3.
- Sebaliknya, pada pasien tanpa spider nevi (N), jumlah kasus menurun dari stadium 3 ke stadium 1.

4. **Edema (pembengkakan pada tungkai bawah)**

- Jumlah pasien tanpa edema (N) menurun dari stadium 3 ke stadium 1.
- Namun, jumlah pasien dengan edema (Y) dan edema berat (S) meningkat secara signifikan pada stadium 3, menunjukkan eskalasi gejala edema seiring dengan kemajuan penyakit.
    -



Kesimpulan

- Sirosis hati merupakan penyakit serius dengan kerusakan hati yang menyebabkan fungsi hati tidak bekerja optimal dan berdampak fatal
- Terdapat 3 stadium sirosis hati : kompensasi, dekompensasi, dan sirosis dekompensata
- Penelitian ini menggunakan algoritma machine learning Random Forest dan Decision Tree dengan platform Spark
- Random Forest menunjukkan performa lebih baik dengan akurasi 95.39%, sedangkan Decision Tree 91.28% sehingga lebih efektif untuk klasifikasi stadium sirosis hati.
- Machine learning memiliki potensi besar dalam mendukung diagnosis medis

Terima Kasih