UJIAN AKHIR SEMESTER

BIG DATA & PREDICT LANJUT

PREDIKSI PENYAKIT HIPERTENSI MENGGUNAKAN 4 METODE

Dosen Pengampu:

Theopilus Bayu Sasongko, S.Kom, M.Eng

Kelompok 5:

ABISAKHA SAIF ALFATH	22.11.5285
DENNITA NOOR FEBIANTY	22.11.4640
MUHAMAD FAHRUROZI	22.11.5132
MUHAMMAD FARREL R A	22.11.5269
NONA ADINDA ARIANA	22.11.5302
HAMZAH SHAFA T.	22.11.5317

Link Launchipad:

https://launchinpad.com/project/prediksi-penyakit-hipertensi-menggunakan-4-metode-gradient-boosting-random-forest-svm-dan-logistic-regresion-779c519

Link GitHub:

https://github.com/abisakha/BDPAL-prediksi-penyakit-hipertensi

PROGRAM STUDI S1 INFORMATIKA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS AMIKOM YOGYAKARTA

2024

DAFTAR ISI

BAB I PENDAHULUAN

1.1 Latar Belakang

Hipertensi merupakan kondisi medis yang ditandai oleh tekanan darah sistolik dan diastolik yang tinggi, yaitu di atas 140/90 mmHg. Penyakit ini sering disebut dengan *"Silent Killer"* karena penyakit ini tidak menunjukan gejala pada sebagian penderita hingga terjadinya komplikasi serius. Hipertensi secara perlahan merusak berbagai organ dalam tubuh, seperti ginjal, jantung, mata, dan otak. Hal tersebut membuat hipertensi menjadi faktor utama penyebab penyakit jantung, gagal ginjal kronis, serta stroke. Menurut data dari Organisasi Kesehatan Dunia (WHO) 2023 diperkirakan 1,28 miliar orang dewasa dalam rentang usia 23-79 tahun di seluruh dunia menderita hipertensi, dengan ⅔ dari data tersebut merupakan penderita yang tinggal di negara berpenghasilan rendah dan menengah. Selain itu, menurut data Kementrian Kesehatan (Kemenkes) 2023 penderita hipertensi di Indonesia mencapai 34,1% atau sekitar 70 juta penduduk. Hal tersebut mengindikasikan 1 dari 3 orang Indonesia menderita hipertensi. Gejala utama yang dirasakan oleh penderita hipertensi dibagi menjadi 2 tipe, yaitu penderita hipertensi ringan dan hipertensi berat. Penderita hipertensi ringan memiliki gejala sakit kepala, nyeri dada, mimisan, dan sesak napas. Sedangkan, penderita hipertensi berat memiliki gejala nyeri pada dada, gemetar tidak terkendali pada otot atau tremor otot, dan mual [1].

Pada saat ini dengan berkembangnya teknologi dan berbagai kecerdasan buatan, menjadi salah satu inovasi utama dalam mendukung berbagai bidang, termasuk sektor kesehatan. *Machine learning* memiliki kemampuan untuk menganalisa data dalam jumlah besar, mempelajari pola tersembunyi, dan memberikan prediksi yang akurat. Hal tersebut meningkatkan kualitas layanan kesehatan, dan membantu dalam memprediksi penyakit. Dalam penerapannya, *Machine Learning* membantu memangkas waktu yang dibutuhkan dalam proses diagnostik medis dengan kemampuan algoritmanya dalam mempelajari pola dari data riwayat pasien dan data laboratorium [2]. Hal tersebut memudahkan staf medis dalam mengidentifikasi dan mendiagnosa penyakit secara akurat dan efisien. Machine learning memiliki berbagai metode dan algoritma dalam menganalisa data dan membuat prediksi. Berdasarkan teknik pembelajarannya, tipe-tipe machine learning dapat dibedakan menjadi supervised learning, unsupervised learning, semi supervised learning dan reinforcement learning [3].

Pemilihan topik ini bertujuan untuk membangun model machine learning yang memprediksi penyebab hipertensi, terdapat beberapa tujuan utama yang ingin dicapai. Salah satu tujuannya adalah mengidentifikasi dan memahami faktor-faktor risiko yang berkontribusi terhadap hipertensi. Melalui analisis data kesehatan, model ini dapat mengungkap pola dan hubungan antara berbagai variabel, seperti usia, pola makan, aktivitas fisik, dan riwayat kesehatan keluarga, sehingga mendukung pengembangan strategi pencegahan yang lebih efektif. Selain itu, model ini bertujuan untuk mendukung pencegahan dini dengan memprediksi kemungkinan seseorang mengembangkan hipertensi, sehingga intervensi seperti perubahan gaya hidup, peningkatan aktivitas fisik, perbaikan pola makan, dan pengelolaan stres dapat dilakukan lebih awal. Model ini juga

berpotensi meningkatkan kesadaran masyarakat tentang risiko hipertensi dengan memberikan informasi yang jelas dan berbasis data, mendorong individu untuk menjaga kesehatan dan melakukan pemeriksaan rutin.

Beberapa penelitian yang menggunakan metode random forest dalam memprediksi penyakit hipertensi sebelumnya. Seperti Penelitian yang dilakukan oleh Purwono Purwono, Pramesti Dewi, Sony Kartika Wibowo, Bala Putra Dewa pada tahun 2022 dengan nilai akurasi ketepatan prediksi yang diperoleh sebesar 85% [4]. Adapun penelitian yang dilakukan oleh Supriyono, Erida, Fadila dengan pada tahun 2022 dengan nilai akurasi sebesar 85% pada proses pelatihan dan 91,89 % pada proses pengujian [5].

Projek ini bertujuan untuk mengidentifikasi faktor-faktor signifikan yang berkontribusi terhadap perkembangan hipertensi. Penelitian sebelumnya telah menunjukkan bahwa pendekatan berbasis data ini dapat memberikan wawasan yang lebih mendalam mengenai interaksi kompleks antara berbagai faktor risiko. Dengan menerapkan 4 metode algortima *Machine learning* dalam memprediksi penyebab hipertensi dengan menganalisis dataset yang relevan. diharapkan dapat memberikan informasi yang berguna bagi tenaga kesehatan dalam upaya pencegahan dan pengelolaan hipertensi secara lebih efektif. Selain itu, penelitian ini juga bertujuan untuk memperkaya literatur mengenai penggunaan teknik analisis data modern dalam bidang kesehatan masyarakat.

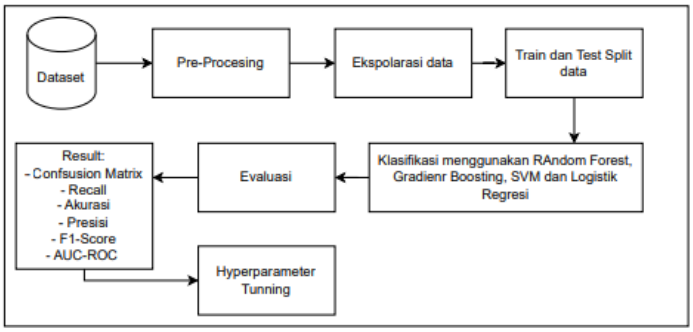
1.2 Rumusan Masalah

1. Bagaimana memastikan kualitas dataset yang digunakan, termasuk memeriksa tipe data, nilai null, duplikasi data, dan outlier, agar data siap digunakan untuk modeling?
2. Bagaimana cara menganalisis dataset melalui visualisasi seperti pie chart, box plot, histogram, dan scatter plot untuk memahami distribusi dan pola hubungan antarvariabel?
3. Fitur apa yang paling relevan untuk digunakan dalam modeling machine learning agar menghasilkan performa terbaik?
4. Model machine learning apa yang memberikan performa terbaik dalam klasifikasi dataset, berdasarkan metrik seperti akurasi, F1-score, precision, recall, dan AUC-ROC?
5. Bagaimana perbandingan performa model seperti Random Forest, Gradient Boosting, Support Vector Machines, dan Logistic Regression dalam menyelesaikan masalah klasifikasi?
6. Bagaimana mengevaluasi model yang dihasilkan menggunakan confusion matrix, akurasi, F1-score, precision, recall, dan AUC-ROC curve untuk memastikan keandalan model?
7. Bagaimana mengatasi ketidakseimbangan kelas pada dataset menggunakan metode seperti SMOTE, dan bagaimana pengaruhnya terhadap performa model?

8. Bagaimana penerapan tuning hyperparameter menggunakan cross-validation pada model seperti Random Forest dan Gradient Boosting dapat meningkatkan performa model?
9. Apakah terdapat peningkatan signifikan dalam metrik evaluasi sebelum dan setelah tuning hyperparameter?
10. Apa saja karakteristik model terbaik yang memberikan hasil paling optimal?

1.3 Metode Penelitian

Tahap metode penelitian merupakan tahap yang mencakup alur atau langkah-langkah yang terencana dan logis dari awal hingga akhir untuk memastikan bahwa hasil penelitian dapat dipercaya dan valid. Dengan menggunakan bahasa pemrograman phyton dan software Google Collaboratory. Tahap penelitian memiliki alur atau langkah-langkah pada diagram alir seperti ditunjukkan pada gambar 1.



Gambar1.3 Diagram Alir Penelitian

BAB II PREPROCESING DATA & EKSPLORASI DATA

2.1 Deskripsi Dataset

Dataset merupakan kumpulan data yang terorganisir dalam format tertentu dan digunakan dalam analis, pelatihan model, penelitian dan sebagai informasi. Dataset Hypertension risk merupakan dataset yang digunakan dalam penelitian ini. Data ini diambil dari datset kaggle: https://www.kaggle.com/datasets/khan1803115/hypertension-risk-model-main/data. Dataset ini terdiri dari atribut demografi dan kesehatan yang ditujukan untuk memprediksi risiko hipertensi dan memiliki 13 fitur, 12 variabel independen dan 1 variabel dependen yang digunakan untuk label kelas yang digunakan untuk memprediksi hipertensi. Dibawah ini rincian atribut yang digunakan dalam penelitian ini, lihat tabel 1.

Tabel 1. Keterangan Atribut Dataset

No	Atribut	*Keterangan*	*Jenis Data*
1	male	Jenis kelamin individu, diindikasikan dengan nilai biner (1 untuk pria, 0 untuk wanita)	int64
2	age	Usia individu dalam tahun	int64
3	currentSmoker	Status merokok saat ini, diindikasikan dengan nilai biner (1 untuk merokok, 0 untuk tidak merokok)	int64

4	cigsPerDay	Jumlah rokok yang dikonsumsi per hari	float64
5	BPMeds	Penggunaan obat-obatan untuk mengatur detak jantung atau tekanan darah, diindikasikan dengan nilai biner (1 untuk pengguna, 0 untuk tidak pengguna)	float64
6	diabetes	Status diabetes individu, diindikasikan dengan nilai biner (1 untuk individu dengan diabetes, 0 untuk individu tanpa diabetes)	int64
7	totChol	Kadar total kolesterol dalam darah (dalam satuan mg/dL)	float64
8	sysBP	Tekanan darah sistolik, yaitu tekanan darah yang diukur saat jantung berkontraksi (dalam satuan mmHg)	float64
9	diaBP	Tekanan darah diastolik, yaitu tekanan darah yang diukur saat jantung beristirahat di antara dua denyut (dalam satuan mmHg)	float64
10	BMI	Indeks Massa Tubuh, yang dihitung berdasarkan rasio antara berat badan (kg) dan kuadrat tinggi badan (m^2)	float64
11	heartRate	Denyut jantung per menit	float64
12	glucose	Kadar glukosa darah (dalam satuan mg/dL)	float64
13	Risk	Skor risiko keseluruhan yang menggambarkan kemungkinan individu mengalami penyakit hipertensi	int64

Tabel 1 merupakan atribut datset yang terdiri dari 13 atribut yang meliputi male, age, currentSmoker, cigsPerDay, BPMeds, diabetes, totChol, sysBP, diaBP, BMI, heartRate, glucose,Risk

2.2 Pre-Prosesing Data

Data mentah yang ada pada datset sebelumnya tidak dapat diproses oleh mesin secara langsung. Perlu dilakukan modifikasi pada data mentah agar menjadi data yang siap diproses oleh mesin [3]. Adapun beberapa tahapan pre-processing dijelaskan dalam sub bab di bawah.

2.2.1 Memeriksa Type Data

Pada tahap ini penting untuk memeriksa type dataset yang terbaca dalam code. Hal ini agar dataset bisa di olah dalam machine learning. Dapat terlihat dalam output yang di hasilkan di bawah, terdapat 13 kolom dengan type data integer dan string. Dalam deskripsi dataset di atas. Tidak adanya fitur dalam dataset yang bertype string. Fitur ini berubah di karenakan adanya data kosong atau *Missing Value* dalam fitur. Yang membuat type data fitur berubah menjadi string. Perlu di lakukan pengubahan type data agar data kosong atau null dapat terhitung dan fitur dapat di olah machine learning.

```
[6]  # Menampilkan tipe data setiap kolom
     df.printSchema()

root
 |-- male: integer (nullable = true)
 |-- age: integer (nullable = true)
 |-- currentSmoker: integer (nullable = true)
 |-- cigsPerDay: string (nullable = true)
 |-- BPMeds: string (nullable = true)
 |-- diabetes: integer (nullable = true)
 |-- totChol: string (nullable = true)
 |-- sysBP: double (nullable = true)
 |-- diaBP: double (nullable = true)
 |-- BMI: string (nullable = true)
 |-- heartRate: string (nullable = true)
 |-- glucose: string (nullable = true)
 |-- Risk: integer (nullable = true)
```

Gambar 2.2.1 Type Data setiap Fitur

2.2.2 Mengubah Type Data

Pada langkah berikut ini merupakan langkah untuk mengubah type data menjadi numeric. Seperti penjelasan pada tahap sebelumnya. Type data string di ubah menjadi numeric. Dapat terlihat dari gambar di bawah, Kode tersebut berfungsi untuk mengubah nilai yang kosong atau nilai yang tidak valid menjadi nilai 0. Kemudian nilai 0 tersebut dirubah tipe datanya menjadi float. Kode tersebut tidak memiliki hasil output dikarenakan perintah yang diberikan hanya mengubah nilai dan tipe data, sehingga tidak memiliki output. Contohnya, pada kolom 'totChol' nilai " " (kosong) diubah menjadi nilai 0. Kemudian menggantinya menjadi tipe data float.

```
[8]  from pyspark.sql.functions import col, when

     # Mengubah nilai kosong atau tidak valid menjadi 0, lalu mengonversinya ke float
     df = df.withColumn("cigsPerDay", when(col("cigsPerDay") == "", None).otherwise(col("cigsPerDay").cast("float")))
     df = df.withColumn("BPMeds", when(col("BPMeds") == "", None).otherwise(col("BPMeds").cast("float")))
     df = df.withColumn("totChol", when(col("totChol") == "", None).otherwise(col("totChol").cast("float")))
     df = df.withColumn("BMI", when(col("BMI") == "", None).otherwise(col("BMI").cast("float")))
     df = df.withColumn("heartRate", when(col("heartRate") == "", None).otherwise(col("heartRate").cast("float")))
     df = df.withColumn("glucose", when(col("glucose") == "", None).otherwise(col("glucose").cast("float")))
```

Gambar 2.2.2 Mengubah Type Data

2.2.3 Memeriksa Nilai Null

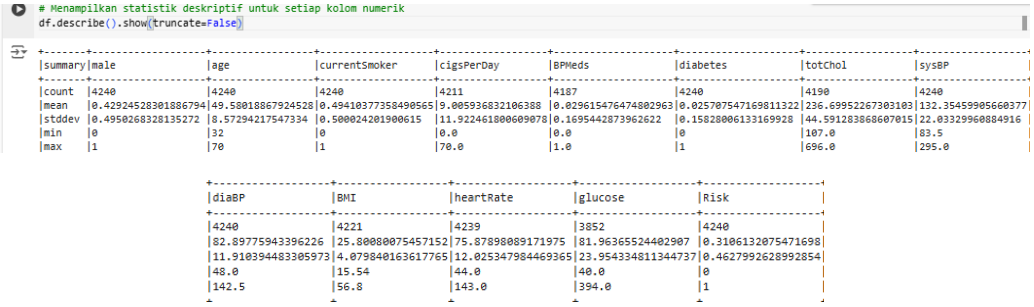
Pada tahap ini, setelah di lakukan mengubah type data. Kemudian dilakukan pengecekan nilai null pada setiap fitur yang ada. Hal ini dilkukan agar data menjadi lebih berkualitas dan tidak menyebabkan bias pada model yang akan di latih. Dari hasil gambar di bawah, terlihat dari 13 fitur ada beberapa fitur yang memiliki nilai null. Dengan nilai null terbanyak pada fiitur Glucose sebanyak 388. Dan kolom lainnya yang nilainya nullnya berada di bawah 100.

```
df.select([count(when(isnan(c) | col(c).isNull(), c)).alias(c) for c in df.columns]).show()

+----+---+-------------+----------+------+--------+-------+-----+-----+---+---------+-------+----+
|male|age|currentSmoker|cigsPerDay|BPMeds|diabetes|totChol|sysBP|diaBP|BMI|heartRate|glucose|Risk|
+----+---+-------------+----------+------+--------+-------+-----+-----+---+---------+-------+----+
|   0|  0|            0|        29|    53|       0|     50|    0|    0| 19|        1|    388|   0|
+----+---+-------------+----------+------+--------+-------+-----+-----+---+---------+-------+----+
```

Gambar 2.2.3 Memeriksa nilai Null

2.2.4 Menampilkan Summary

Pada tahap menampilkan Hasil summary statistik untuk memberikan gambaran awal tentang distribusi data dalam dataset. Kolom count menunjukkan jumlah nilai non-null untuk setiap variabel, di mana beberapa kolom seperti cigsPerDay, BPMeds, dan glucose memiliki data yang hilang karena jumlahnya lebih kecil dari total (4240). mean dan stddev memberikan rata-rata dan standar deviasi untuk setiap kolom, mencerminkan variabilitas dalam data, misalnya rata-rata usia adalah sekitar 49,58 tahun dengan standar deviasi 8,57. Kolom seperti totChol memiliki nilai maksimum 696, yang tampaknya outlier dibandingkan rata-ratanya (236,7). Nilai minimum, seperti sysBP (83,5), menunjukkan rentang nilai data yang besar di beberapa kolom. Kehadiran nilai-nilai ekstrim seperti pada BMI (maksimum 56,8) dan glucose (maksimum 394) menyoroti perlunya pengolahan data lebih lanjut, termasuk menangani outlier dan nilai hilang. Variabel target Risk memiliki rata-rata 0,31, menunjukkan ketidakseimbangan data antara kelas risiko.



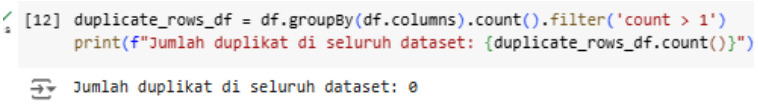
```
# Menampilkan statistik deskriptif untuk setiap kolom numerik
df.describe().show(truncate=False)
```

```
+-------+------------------+------------------+-------------------+------------------+------------------+-------------------+------------------+------------------+
|summary|male              |age               |currentSmoker      |cigsPerDay        |BPMeds            |diabetes           |totChol           |sysBP             |
+-------+------------------+------------------+-------------------+------------------+------------------+-------------------+------------------+------------------+
|count  |4240              |4240              |4240               |4211              |4187              |4240               |4190              |4240              |
|mean   |0.4292452830188679|49.58018867924528 |0.4941037735849057 |9.005936832106388 |0.02961547647482963|0.02570754716981322|236.69952267303103|132.35459905660377|
|stddev |0.4950268328135272|8.572942175473344 |0.5000242019006015 |11.922461800609078|0.16954428739626222|0.15828006133169928|44.591283868607015|22.033299608849164|
|min    |0                 |32                |0                  |0.0               |0.0               |0                  |107.0             |83.5              |
|max    |1                 |70                |1                  |70.0              |1.0               |1                  |696.0             |295.0             |
+-------+------------------+------------------+-------------------+------------------+------------------+-------------------+------------------+------------------+

+------------------+------------------+------------------+------------------+-------------------+
|diaBP             |BMI               |heartRate         |glucose           |Risk               |
+------------------+------------------+------------------+------------------+-------------------+
|4240              |4221              |4239              |3852              |4240               |
|82.89775943396226 |25.800800754571527|75.87898089171975 |81.96365524402907 |0.3106132075471698 |
|11.910394483305973|4.079840163617765 |12.025347984469365|23.954334811344737|0.46279926289928544|
|48.0              |15.54             |44.0              |40.0              |0                  |
|142.5             |56.8              |143.0             |394.0             |1                  |
+------------------+------------------+------------------+------------------+-------------------+
```

Gambar 2.2.4 Menampilkan Summary

2.2.5 Memeriksa Nilai Duplikat

Pada tahap ini dilakukan pengecekan nilai atau data duplikat dalam datset. Hal ini penting untuk mencegah *overfiting* dalam model akibat banyaknya data duplikat dalam datset. Hal ini dianalisis menggunakan perintah groupBy pada semua kolom dataset, diikuti dengan penghitungan jumlah kemunculan baris yang sama. Jika ada baris yang muncul lebih dari sekali, perintah ini akan mencatatnya. Dari output tersebut, hasil menunjukkan bahwa semua baris dalam dataset bersifat unik, dengan jumlah duplikat sebanyak 0.



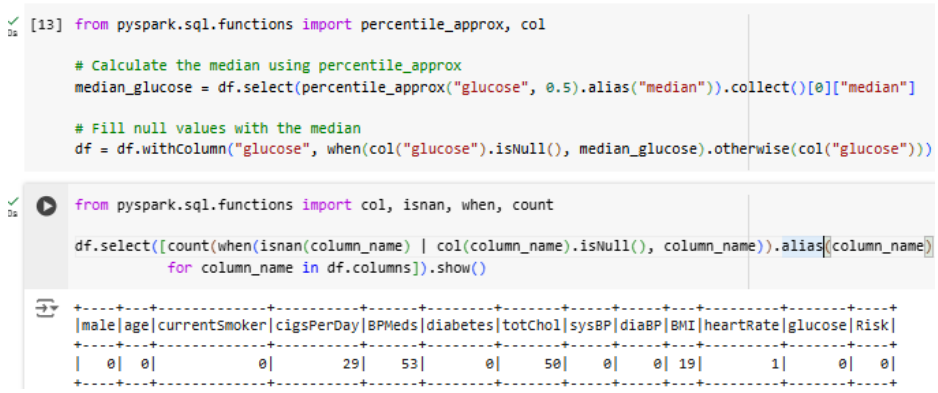
```
[12] duplicate_rows_df = df.groupBy(df.columns).count().filter('count > 1')
     print(f"Jumlah duplikat di seluruh dataset: {duplicate_rows_df.count()}")

Jumlah duplikat di seluruh dataset: 0
```

Gambar 2.2.5 Memeriksa Data Duplikat

2.2.6 Imputasi Nilai Null memakai Median

Pada tahap ini dilakukan pengisian data kosong menggunakan teknik *Imputation with Median.* Teknik ini digunakan karena data yang kosong pada kolom Glucose sangat banyak melebihi 2% dari keseluruhan data dalam dataset. Apabila digunakan teknik drop maka akan berpengaruh pada model dan menyebabkan bias pada model. Teknik Imputasi menggunakan nilai median ialah metode efektif untuk menangani data yang hilang, khususnya pada dataset distribusi tidak normal, karena nilai median lebih tahan terhadap

pengaruh nilai ekstrim dibandingkan rata-rata(mean)[6]. Pada gambar di bawah dapat terlihat data kosong dalam glucose sudah tidak ada. Hal ini karena nilai kosng tersebut di isi dengan nilai median dari fitur tersebut.



```
[13] from pyspark.sql.functions import percentile_approx, col

     # Calculate the median using percentile_approx
     median_glucose = df.select(percentile_approx("glucose", 0.5).alias("median")).collect()[0]["median"]

     # Fill null values with the median
     df = df.withColumn("glucose", when(col("glucose").isNull(), median_glucose).otherwise(col("glucose")))
```

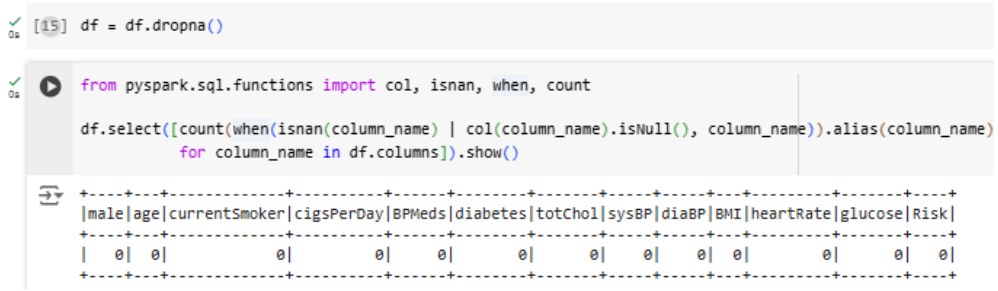
```
from pyspark.sql.functions import col, isnan, when, count

df.select([count(when(isnan(column_name) | col(column_name).isNull(), column_name)).alias(column_name)
           for column_name in df.columns]).show()
```

```
+----+---+-------------+----------+------+--------+-------+-----+-----+---+---------+-------+----+
|male|age|currentSmoker|cigsPerDay|BPMeds|diabetes|totChol|sysBP|diaBP|BMI|heartRate|glucose|Risk|
+----+---+-------------+----------+------+--------+-------+-----+-----+---+---------+-------+----+
|   0|  0|            0|        29|    53|       0|     50|    0|    0| 19|        1|      0|   0|
+----+---+-------------+----------+------+--------+-------+-----+-----+---+---------+-------+----+
```

Gambar 2.2.6 Memeriksa Data Duplikat

2.2.7 Imputasi Nilai Null dengan Drop Missing Value

Pada tahap selanjutnya, setelah melakukan pengsian nilai kosng pada glucose. Dilakukan pengisian data kosong pada kolom lainnya yang tersisa. Tahap ini menggunakan teknik preprocessing yaitu menghapus atau mendrop data kosong. Hal ini dilakukan setelah dilakukan teknik imputasi menggunakan median, data yang kosong berjumlah di bawah 2%. Teknik drop digunakan karena jumlah data yang hilang kurang dari 5%, karena tingkat kehilangan data yang rendah ini tidak secara signifikan mempengaruhi estimasi hasil penelitian atau meningkatkan risiko bias [7]. Dapat terlihat dalam gambar di bawah, setelah dilakukan pengisian dan penghapusan nilai kosong. Data yang kosong dalam dataset ini telah hilang.



```
[15] df = df.dropna()
```

```
from pyspark.sql.functions import col, isnan, when, count

df.select([count(when(isnan(column_name) | col(column_name).isNull(), column_name)).alias(column_name)
           for column_name in df.columns]).show()
```

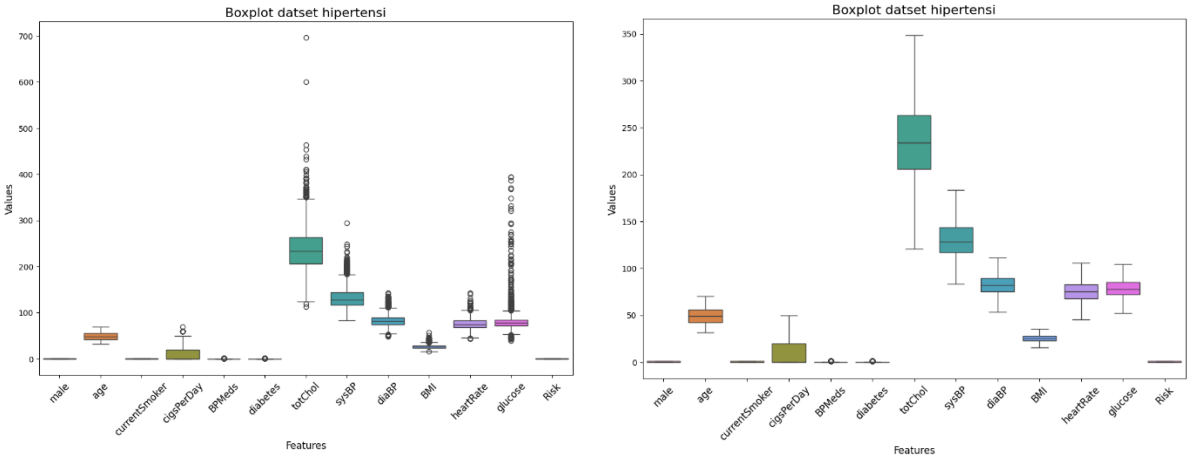
```
+----+---+-------------+----------+------+--------+-------+-----+-----+---+---------+-------+----+
|male|age|currentSmoker|cigsPerDay|BPMeds|diabetes|totChol|sysBP|diaBP|BMI|heartRate|glucose|Risk|
+----+---+-------------+----------+------+--------+-------+-----+-----+---+---------+-------+----+
|   0|  0|            0|         0|     0|       0|      0|    0|    0|  0|        0|      0|   0|
+----+---+-------------+----------+------+--------+-------+-----+-----+---+---------+-------+----+
```

Gambar 2.2.7 Memeriksa Data Duplikat

2.2.8 Remove Outlier

Pada tahap ini dilakukan remove outlier. hal ini di lakukan untuk meningkatkan akurasi, mengurangi bias dalam statistik, meningkatkan kinerja model, mencegah overfitting, dan mengurangi noise. Dapat terlihat dalam gambar di bawah, setelah di lakukan penghapusan ooutlier. Data menjadi lebih bersih dari data pencilan.

Sebelum: Sesudah:

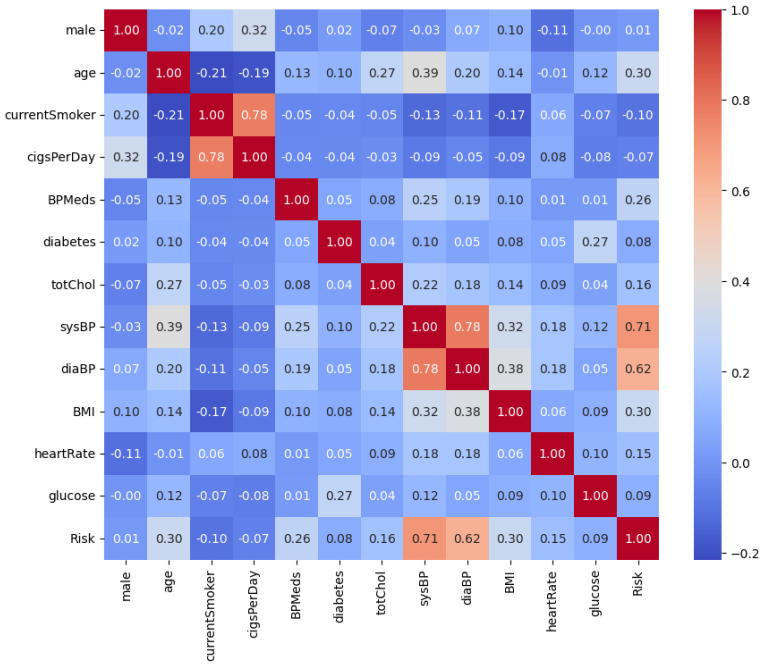


Gambar 2.2.8 Remove Outlier

2.2.9 Visualisasi Korelasi Matrix

Pada tahap ini dilakukan visualisasi korelasi matrix untuk melihat hubungan antar variabel dalam dataset dengan intensitas korelasi yang direpresentasikan dalam warna. Nilai korelasi positif tertinggi ditemukan antara sysBP dan diaBP (0.78), menunjukkan bahwa tekanan darah sistolik (sysBP) dan diastolik (diaBP) memiliki hubungan linier yang sangat kuat. Hubungan yang cukup signifikan juga terlihat antara sysBP dan Risk (0.71), serta diaBP dan Risk (0.62), yang menunjukkan bahwa peningkatan tekanan darah terkait erat dengan peningkatan risiko kesehatan. Variabel age memiliki korelasi sedang dengan Risk (0.30), menandakan bahwa usia juga berkontribusi terhadap risiko. Namun, sebagian besar variabel lainnya memiliki korelasi yang rendah hingga moderat terhadap Risk, seperti BMI, glucose, dan totChol. Hal ini menunjukkan bahwa meskipun variabel-variabel tersebut berkontribusi terhadap risiko, dampaknya tidak sekuat variabel tekanan darah. Secara keseluruhan, analisis ini menunjukkan pentingnya tekanan darah dalam prediksi risiko kesehatan.

```
# visualisai data menggunakan heatmap, melihat korelaso antar variable(analisis bivariet )
# Convert PySpark DataFrame to Pandas DataFrame
pandas_df = df.toPandas()

# Calculate correlation matrix using the Pandas DataFrame
corr_matrix = pandas_df.corr()

plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.show()
```



Gambar 2.2.9 Visualisasi Korelasi Matrix

2.3 Eksplorasi Data

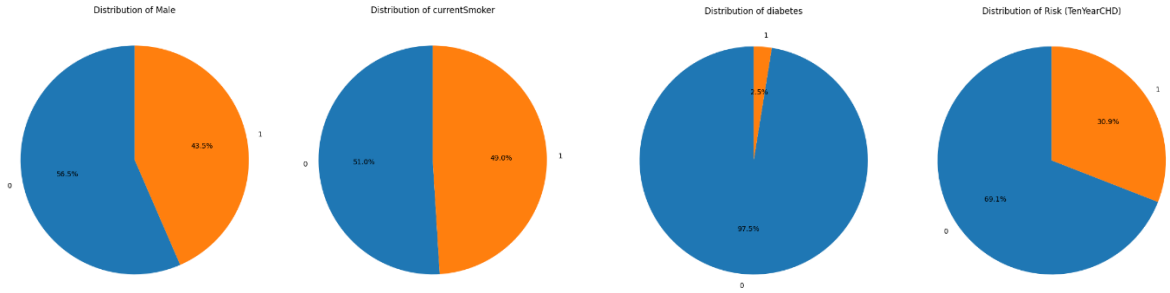
Eksplorasi data adalah untuk memahami karakteristik dataset, termasuk distribusi, hubungan antar variabel, dan keberadaan outlier atau data yang hilang. Proses ini membantu mengidentifikasi pola, tren, dan anomali yang dapat memengaruhi analisis atau model machine learning. Dengan eksplorasi data, kita dapat menentukan langkah preprocessing yang diperlukan dan memilih fitur yang relevan untuk analisis lebih lanjut.

2.3.1 Visualisasi Pie Chart

Pada gambar di bawah, memvisualkan 4 kolom, yaitu male CurrentSmoker, Diabetes, dan Risk. Pada kolom male distribusi data female lebih banyak di bandingkan male. Dengan distribusi female 54,5% dan male 43,5%. Pada fitur CurrentSmoker distribusi data cenderung seinmbang dengan sedikit perbedaan yang sangat sedikit yaitu sekitar 1% pada kedua data.

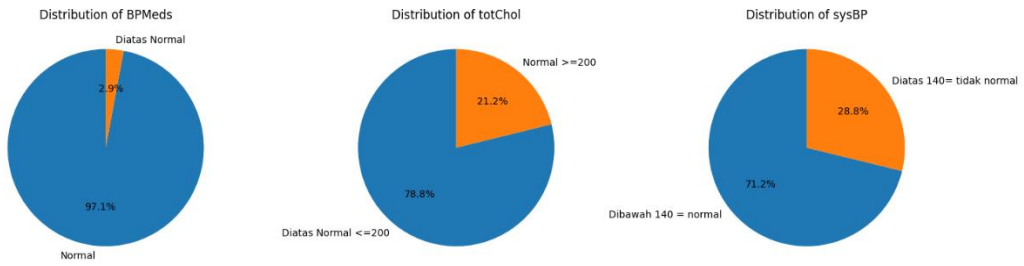
Pada fitur Diabetes, distribusi data sangat tidak seinmbang. Antara seseorang terkena diabates dan tidak. Lebih banyak data seseorang yang tidak terkena diabetes dengan distribusi 97,5% dan terkena diabetes sebanyak 2,5% saja. Sedangkan pada fitur Risk data sedikit tidak seimbang dengan persentase seseorang yang terkena hipertensi sebanyak

30,9% dan seseorang yang normal sebanyak 69,1%. Perlu di lakukan balancing data pada fitur ini. karena fitur ini menjadi label dalam model.



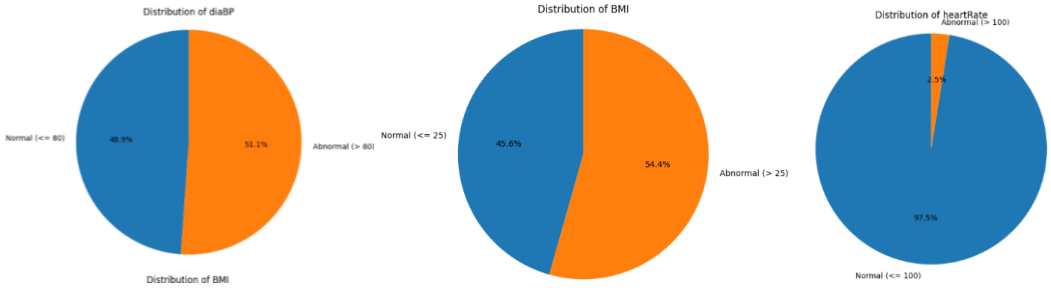
Gambar 2.3.1.1 Visualisasi Male, CurrentSmoker, Diabetes, Risk.

Pada gambar 2.3.1.2 terdapat 3 fitur, yaitu BPMedis, totChol, dan sysBp. Pada fitur BPMedis ditribusi lebih banyak ke normal di bandingkan di atas normal. Dengan persentase normal sebanyak 97.1% dan di atas normal sebanyak 2,9%. Pada fitur totChol distribusi data cenderung lebih banyak ke normal atau di bawah 200. Hal ini menandakan kebanyakan pada datset ini lebih banyak seseorang normal di bandingkan kadar kolestrol di atas normal. Pada fitur sysBp atau tekanan sistolik. Distribusi fitur ini secara keleuruhan tidak seimbang. Lebih banyak seseorang yang tekanan darah sistoliknya normal dengan persentase 71,2% dan tekanan sistolik di atas normal sebanyak 28,8%.



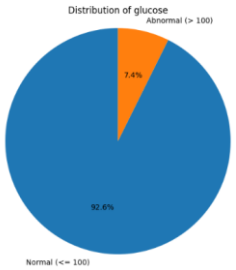
Gambar 2.3.1.2 Visualisasi BPMedis, totChol, dan sysBp

Pada gambar 2.3.1.3 terdapat 3 fitur yaitu diaBP, BMI, dan Heartrate. Pada fitur diaBP distribusinya lebih besar ke abnormal daripada normal yang dimana nilai persentase nya sebesar 51,1% untuk abnormal sedangkan yang normal sebesar 48,9%. Pada fitur BMI distribusinya masih cenderung lebih ke abnormal daripada yang normal yang dimana persentasenya sebesar 54,4% dan yang normal sebesar 45,6%. Dan yang terakhir pada fitur heartrate distribusinya mayoritas normal dengan persentase 97,5% dan yang abnormal hanya 2,5%.



Gambar 2.3.1.3 Visualisasi diaBP, BMI, dan Heartrate

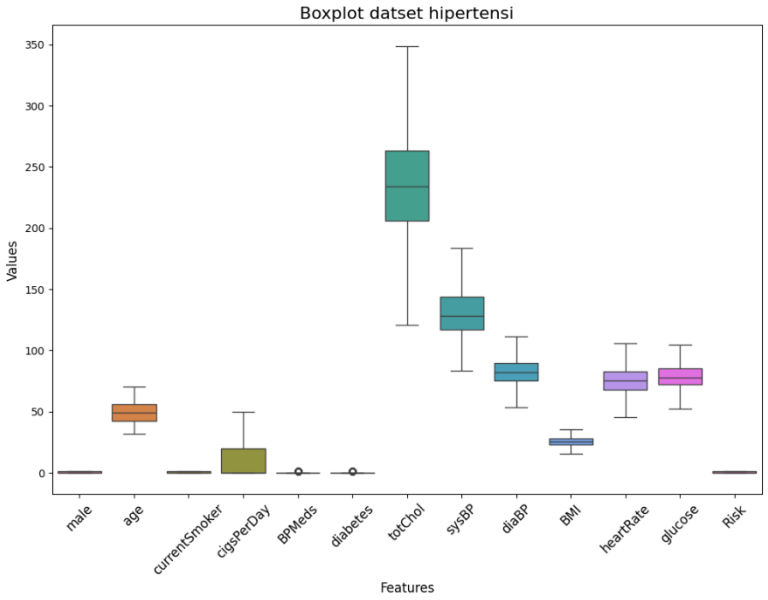
Pada gambar 2.3.1.4 terdapat 3 fitur, yaitu Gluscose, Diabetes, dan Risk. Pada fitur Glucose disribusi lebih banyak ke normal di bandingkan di atas normal. Dengan persentase normal sebanyak 92.6% dan di atas normal sebanyak 7,4%.



Gambar 2.3.1.4 Visualisasi Gluscose, Diabetes, dan Risk

2.3.2 Visualisasi Box Plot

Pada tahap ini melakukan visualisasi box plot. Terlihat pada gambar di bawah tidak adanya data pencilan. Data terlihat bersih dengan scala yang terukur. Hal ini di karenakan telah di lakukan remove outlier pada tahap pre procesing di atas.

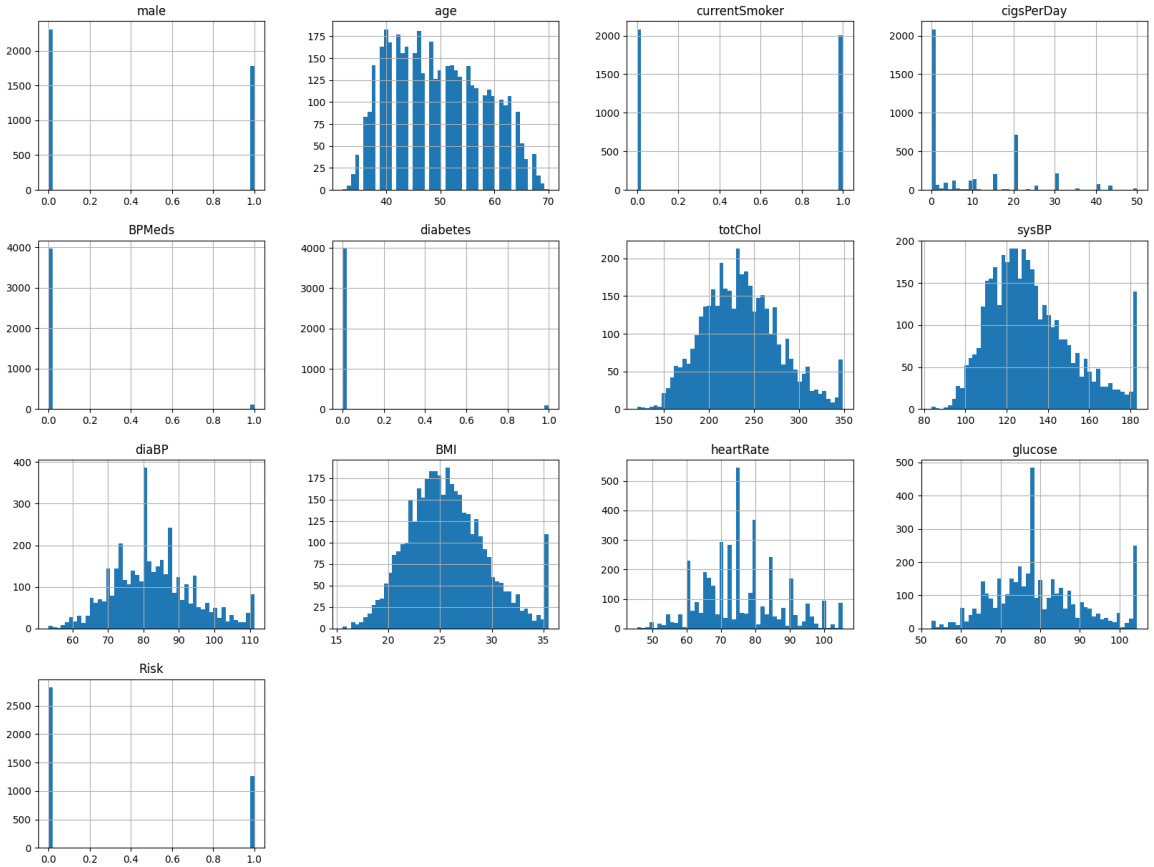


Gambar 2.3.2 Visualisasi Box Plot

2.3.3 Visualisasi Histogram

Gambar histogram di atas menunjukkan distribusi data untuk berbagai fitur dalam dataset. Beberapa fitur, seperti male, currentSmoker, BPMeds, diabetes, dan Risk, memiliki distribusi biner (0 dan 1), yang menunjukkan bahwa mereka adalah variabel kategori atau indikator. Fitur-fitur lain seperti age, totChol, sysBP, diaBP, dan BMI menunjukkan distribusi yang menyerupai distribusi normal dengan sedikit variasi, meskipun ada beberapa outlier pada fitur seperti sysBP dan BMI. Fitur cigsPerDay dan glucose menunjukkan distribusi yang lebih tersebar dengan puncak di nilai tertentu, menunjukkan bahwa ada kelompok populasi dengan kebiasaan atau kondisi kesehatan tertentu.

Pada analisis gambar di bawah menunjukkan adanya variasi yang signifikan dalam skala dan distribusi data di setiap fitur. Hal ini menunjukkan pentingnya normalisasi atau standarisasi untuk model yang sensitif terhadap skala, terutama untuk fitur numerik seperti cigsPerDay, totChol, dan sysBP. Outlier yang terlihat dalam histogram juga perlu ditangani, karena mereka dapat memengaruhi performa model machine learning. Secara keseluruhan, visualisasi ini membantu dalam mengidentifikasi karakteristik unik dari setiap fitur dan menentukan langkah preprocessing yang tepat.

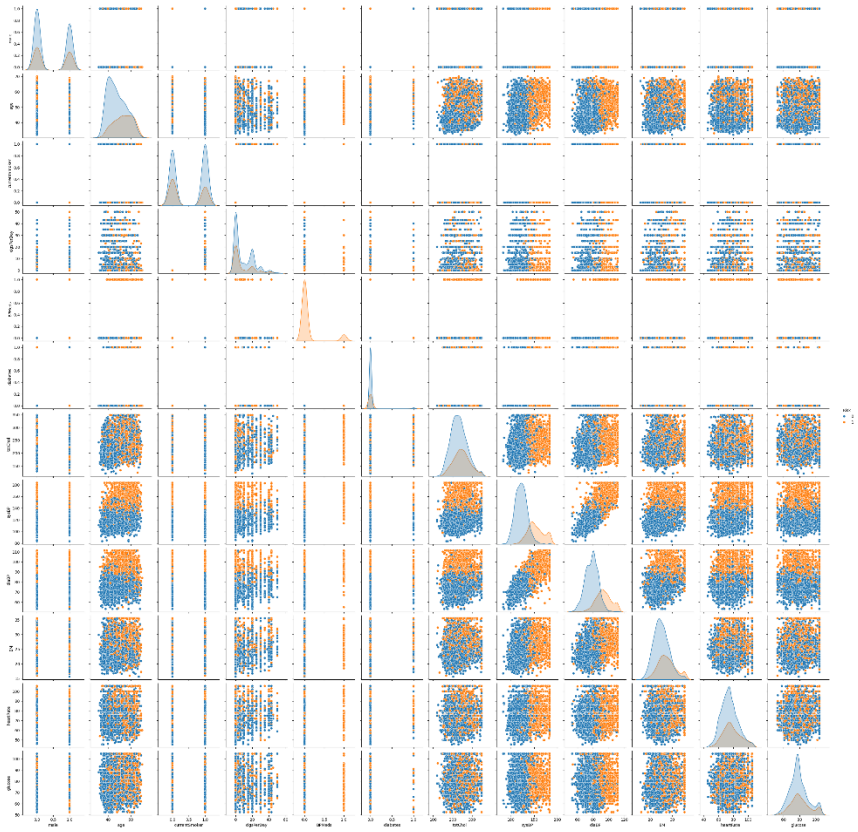


Gambar 2.3.3 Visualisasi Histogram

2.3.4 Visualisasi Scater Plot

Gambar Scater plot menunjukkan distribusi dan hubungan antar fitur dalam dataset, dibedakan berdasarkan variabel target Risk. Fitur dengan distribusi biner, seperti male, currentSmoker, BPMeds, dan diabetes, menunjukkan penyebaran yang seragam antara kategori risiko rendah dan tinggi, sementara beberapa fitur numerik, seperti age, sysBP, dan BMI, menunjukkan pola distribusi yang berbeda antara kedua kategori. Misalnya, age cenderung lebih tinggi pada individu dengan risiko lebih besar, sementara fitur sysBP menunjukkan hubungan positif yang kuat dengan Risk.

Dari scatterplot antar fitur, hubungan linear terlihat lebih jelas antara sysBP dan diaBP, yang menunjukkan korelasi tinggi. Selain itu, fitur seperti totChol dan BMI menunjukkan penyebaran yang relatif seragam tanpa pola yang terlalu jelas, menunjukkan bahwa mereka mungkin kurang berkontribusi secara langsung dalam memprediksi Risk. Pairplot ini membantu dalam mengevaluasi fitur mana yang memiliki hubungan signifikan dengan variabel target dan hubungan antar fitur yang mungkin menunjukkan multikolinearitas, memberikan panduan.



Gambar 2.3.4 Visualisasi Scater plot

2.4 Pemilihan Fitur untuk Modeling

Pemilihan fitur untuk pemodelan bertujuan untuk menentukan variabel-variabel yang relevan dalam memprediksi target Risk. Beberapa fitur seperti age, sysBP, diaBP, dan BMI sangat relevan berdasarkan domain kesehatan, karena tekanan darah, usia, dan obesitas dikenal sebagai faktor utama yang mempengaruhi risiko hipertensi. Fitur glucose dan totChol juga dapat dimasukkan karena kadar glukosa dan kolesterol yang abnormal sering dikaitkan dengan gangguan kardiovaskular.

Fitur biner seperti male, currentSmoker, dan diabetes dapat memberikan informasi tambahan mengenai risiko. Misalnya, perbedaan risiko pada pria dan wanita (male), serta dampak kebiasaan merokok (currentSmoker) dan diabetes, menjadi pertimbangan penting karena variabel ini secara klinis relevan.

Secara keseluruhan fitur dalam datset ini tidak ada yang di hapus. Hal ini di karenakan semua fitur yang ada sangat penting untuk membantu model dalam memprediksi penyakit hipertensi dengan fitur Risk sebagai Label klasifikasi.

BAB III MODEL ALGORITMA, EVALUASI & HYPERTUNNING PARAMETER

3.1 Deskripsi Metode Machine Learning

Dalam upaya memahami mengevaluasi performa algoritma machine learning pada masalah klasifikasi, penelitian ini di ancang dengan pendekatan eksperimental. Pendekatan ini memberikan dasar untuk menganalisa efektivitas model dalam memproses data, memprediksi label, dan mencapai hasil yang optimal. Melalui metodologi yang sistematis, penelitian ini bertujuan untuk memberikan wawasan yang jelas mengenai langkah- langkah yang d ambil.

Penelitian ini menggunakan 4 model algoritma yang berbeda guna mencapai akurasi yang tinggi. Dan predikisi yang lebih akurat dan efektiv. Algoritma pertama yaitu random Forest adalah algoritma ensemble berbasis pohon keputusan yang menggunakan metode bagging untuk membuat kumpulan pohon keputusan yang independen, kemudian menggabungkan hasil voting mayoritas (klasifikasi) atau rata-rata (regresi) untuk meningkatkan akurasi dan mengurangi risiko overfitting. Gradient Boosting, di sisi lain, adalah teknik ensemble yang menggabungkan prediksi dari model-model sederhana (seperti pohon keputusan) secara iteratif, di mana setiap model baru berusaha memperbaiki kesalahan model sebelumnya, sehingga cocok untuk data yang kompleks. Support Vector Machine (SVM) adalah algoritma pembelajaran yang mencari hyperplane optimal untuk memisahkan data dalam ruang fitur, dengan margin maksimum antara kelas, sering digunakan untuk klasifikasi dan regresi. Logistic Regression adalah model statistika yang digunakan untuk memprediksi probabilitas kejadian suatu kelas berdasarkan hubungan linier antara fitur input dan log odds dari output biner, sederhana namun efektif untuk banyak kasus klasifikasi.

3.2 Spliting data

Pada tahap spiliting data ini, dataset ini di bagi menjadi 2 bagian yaitu yaitu label dan fitur. Hal ini dilakukan karena untuk menetukan fitur apa saja yang akan di latih dan di jadikan label dalam klasifikasi model. Dapat terlihat dalam code gambar di bawah, fitur yang di jadikan label ialah fitur risk. Sedangkan pada fitur lainnya seperti male, age, sysBP dan lainnya menjadi fitur yang akan di latih oleh model. Hal ini perlu di lakukan karena untuk memisahkan fitur untuk label dan membuat model tidak kebingungan atau bisa dalam membuat kelas klasifikasi.

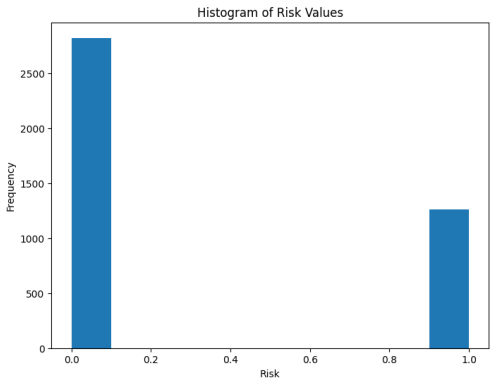
```
from imblearn.over_sampling import SMOTE
from pyspark.ml.feature import VectorAssembler

# Convert PySpark DataFrame to Pandas DataFrame
df_pandas = df.toPandas()
features = ['male', 'age', 'currentSmoker', 'cigsPerDay', 'BPMeds', 'diabetes', 'totChol', 'sysBP', 'diaBP',
            'BMI', 'heartRate', 'glucose']
X = df_pandas[features]
y = df_pandas['Risk']
```

Gambar 3.2 Spliting data

3.2.1 Cek Balancing Data

Pada tahap selanjutnya, setelah melakukan pemisahan fitur dataset. Tahap yang di lakukan ialah melakukan cek balancing data. Balancing data ialah suatu metode yang digunakan untuk memastikan distibusi kelas pada variabel target seimbang. Terutama jika terdapat ketidakseimbangan jumlah antara kkategori tertentu. Dalam kasus ini target biner ialah pada fitur Risk (dengan nilai data fitur ialah 0 dan 1). Dapat terlihat dalam gambar bar chart di bawah. Adanya ketidakseimbangan data. Dimana data 0 lebih banyak dari pada data 1, dengan selisih data hampir setengah. Hal ini perlu di balancing atau di seimbangkan agar nanti ketika model mulai mentraining data, maka model tidak overfitiing yang hanya mengenali data mayortitas saja.

```
import matplotlib.pyplot as plt

print(y.value_counts())


plt.figure(figsize=(8, 6))
plt.hist(y, bins=10)  # Use the 'y' Series directly
plt.xlabel('Risk') # Change x-axis label to 'Risk'
plt.ylabel('Frequency')
plt.title('Histogram of Risk Values') # Change title to reflect 'Risk'
plt.show()
```

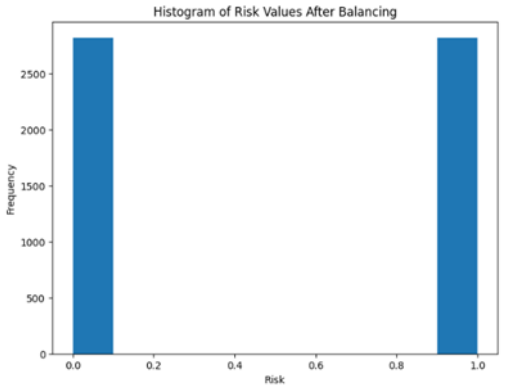


Gambar 3.2.1 cek balancing data

3.2.2 Balancing Data memakai SMOTE dan Spliting Train dan Test

Pada tahap ini dilakukan balacing data menggunakan teknik SMOTE, data yang awalnya tidak seimbang (kelas 0 jauh lebih dominan daripada kelas 1) menjadi lebih seimbang. Hal ini memungkinkan model machine learning untuk belajar secara lebih efektif dari kedua kelas, meningkatkan performa terutama pada kelas minoritas (1). Namun, penting untuk memeriksa apakah data sintetis yang dihasilkan relevan dan tidak memperkenalkan noise tambahan. Setalah di lakukan balancing data maka dataset di bagi menjadi 2 yaitu train dan test. Hal ini di lakukan untuk memisahkan data untuk melatih model dan data untuk menguji model. Dengan pembagian 80% dan 20%.

```
# Apply SMOTE
smote = SMOTE(sampling_strategy='auto', random_state=42)
X_resampled, y_resampled = smote.fit_resample(X, y)

# Convert the resampled data back to pandas DataFrame
df_resampled = pd.DataFrame(X_resampled, columns=features)
df_resampled['Risk'] = y_resampled

# Convert the resampled DataFrame back to Spark DataFrame
df_spark_resampled = spark.createDataFrame(df_resampled)

# 2. Split data into training and testing sets (80% train, 20% test)
train_df, test_df = df_spark_resampled.randomSplit([0.8, 0.2], seed=42)

# 3. Assemble features into a vector
assembler = VectorAssembler(inputCols=features, outputCol="features")
train_df = assembler.transform(train_df)
test_df = assembler.transform(test_df)
```



Gambar 3.2.2 balancing data menggunakan teknik SMOTE

3.3 Random Forest

Random Forest adalah algoritma ensemble berbasis pohon keputusan yang menggunakan beberapa decision tree untuk meningkatkan akurasi prediksi. Dalam penelitian ini setlah di lakukan pelatihan model. Di dapati akurasi sebesar 0.91 (91%) menunjukkan bahwa model dapat memprediksi dengan benar 91% dari total data uji.

```python
# Import libraries
from pyspark.ml.classification import RandomForestClassifier
from pyspark.ml.evaluation import MulticlassClassificationEvaluator, BinaryClassificationEvaluator
from pyspark.ml.feature import VectorAssembler
from pyspark.sql import SparkSession
from imblearn.over_sampling import SMOTE
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from pyspark.sql.functions import col
from sklearn.model_selection import train_test_split

# Initialize Spark session
spark = SparkSession.builder.appName("RandomForestClassifier").getOrCreate()


# 4. Initialize Random Forest Classifier
rf_classifier = RandomForestClassifier(featuresCol="features", labelCol="Risk", numTrees=100, seed=42)

# 5. Fit model with training data
rf_model = rf_classifier.fit(train_df)

# 6. Predict using test data
predictions = rf_model.transform(test_df)

# 7. Evaluate Model
# a. Accuracy
accuracy_evaluator = MulticlassClassificationEvaluator(labelCol="Risk", predictionCol="prediction", metricName="accuracy")
accuracy = accuracy_evaluator.evaluate(predictions)
print(f"Akurasi Model: {accuracy:.2f}")

# b. Confusion Matrix
conf_matrix = predictions.groupBy("Risk", "prediction").count().toPandas()
conf_matrix_pivot = conf_matrix.pivot(index="Risk", columns="prediction", values="count").fillna(0)

# c. Visualize Confusion Matrix
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix_pivot, annot=True, fmt="d", cmap="Blues", xticklabels=["Negative", "Positive"], yticklabels=["Negative", "Positive"])
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.title("Confusion Matrix")
plt.show()



# e. Classification Report
from sklearn.metrics import classification_report

def extract_labels_and_predictions(predictions):
    labels_and_preds = predictions.select("Risk", "prediction").rdd.map(lambda row: (row["Risk"], row["prediction"])).collect()
    y_true, y_pred = zip(*labels_and_preds)
    return y_true, y_pred

y_true, y_pred = extract_labels_and_predictions(predictions)
report = classification_report(y_true, y_pred, target_names=["Negative", "Positive"])
print("\nClassification Report:")
print(report)
```

```
Akurasi Model: 0.91
```

Gambar 3.3 model training dan akurasi

3.3.1 Evaluasi Matrix Confusion

Pada tahap ini di lakukan evaluasi model. Hal ini dilakukan untuk melihat bagaimana model memprediksi data test dengan benar dan salah. Dari gambar di bawah, Confusion Matrix menunjukkan hasil prediksi model terhadap data uji. Berikut adalah nilai-nilai penting:

- True Negative (TN) = 464
  Model memprediksi kelas Negative dengan benar ketika label sebenarnya adalah Negative.
- False Positive (FP) = 60
  Model memprediksi kelas Positive ketika label sebenarnya adalah Negative (kesalahan model).
- True Positive (TP) = 506
  Model memprediksi kelas Positive dengan benar ketika label sebenarnya adalah Positive.
- False Negative (FN) = 32
  Model memprediksi kelas Negative ketika label sebenarnya adalah Positive (kesalahan model).

Gambar 3.3.1 Evaluasi Matrix

3.3.2 Evalusai Akurasi, F-1 Score, Precission dan Recall

Pada gambar di bawah merupakan evaluasi model secara mendatar berdasarkan Classification Report:

- Akurasi: Model memiliki akurasi sebesar 91%, menunjukkan bahwa model mampu memprediksi dengan benar 91% dari total data uji.
- F1-Score: Keseimbangan antara precision dan recall menunjukkan performa yang baik:
  - Kelas Negative: 0.91
  - Kelas Positive: 0.92
- Precision: Kemampuan model memprediksi kelas dengan benar:
  - Kelas Negative: 0.94 (94% prediksi Negative benar)
  - Kelas Positive: 0.89 (89% prediksi Positive benar)
- Recall: Kemampuan model menangkap semua data dari setiap kelas:
  - Kelas Negative: 0.89 (89% data Negative dikenali dengan benar)
  - Kelas Positive: 0.94 (94% data Positive dikenali dengan benar)

Model menunjukkan performa yang seimbang antara kedua kelas dengan Score 0.91 (macro average). Precision sedikit lebih rendah untuk kelas Positive,yang dapat diperbaiki melalui analisis lebih lanjut terhadap false positive.

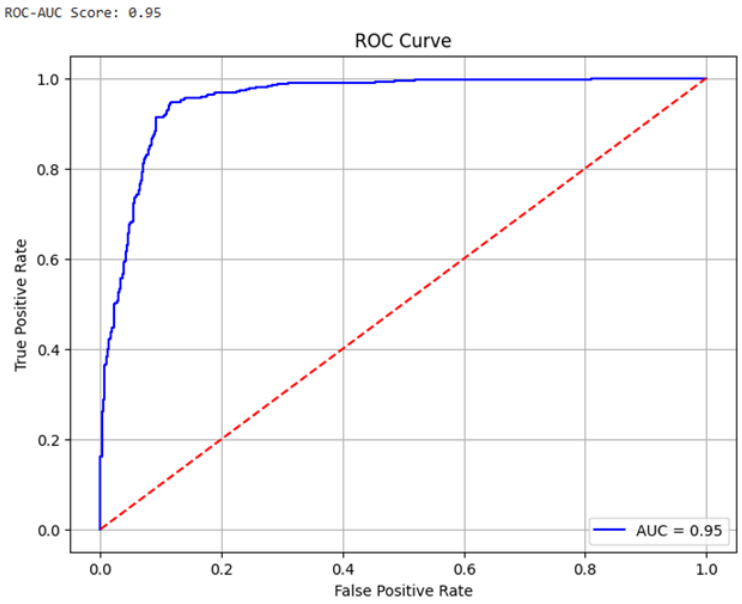
Classification Report:

	precision	recall	f1-score	support
Negative	0.94	0.89	0.91	524
Positive	0.89	0.94	0.92	538
accuracy			0.91	1062
macro avg	0.91	0.91	0.91	1062
weighted avg	0.91	0.91	0.91	1062

Gambar 3.3.2 Evaluasi report

3.3.3 Evaluasi AUC-ROC

Pada tahap ini model menghasilkan evaluasi model menghasilkan ROC-AUC = 0.95, menunjukkan kemampuan yang sangat baik dalam membedakan kelas Positive dan Negative. ROC curve mendekati sudut kiri atas, yang berarti model memiliki True Positive Rate (TPR) tinggi dan False Positive Rate (FPR) rendah. Nilai AUC = 0.95 mengindikasikan bahwa model dapat membedakan kelas dengan akurasi 95%. Performa ini menunjukkan model bekerja optimal, tetapi threshold optimal dapat dievaluasi untuk menyeimbangkan Precision dan Recall lebih lanjut. Model ini cocok untuk diterapkan pada dataset dengan kebutuhan prediksi yang tinggi.



Gambar 3.3.3 Evaluasi AUR-ROC

3.4 Gradient Boosting

Gradient Boosting adalah algoritma ensemble yang membangun model prediktif secara bertahap, dengan menggabungkan model-model sederhana (seperti decision tree) secara berurutan. Setiap model baru mencoba mengurangi kesalahan residu yang dihasilkan oleh model sebelumnya. Dalam peneltian ini di dapati akurasi dari model ini sebesar 0.92 (92%). Angka ini lebih tinggi di bandingkan dengan model algortima random forest dengan selisih 1% saja.

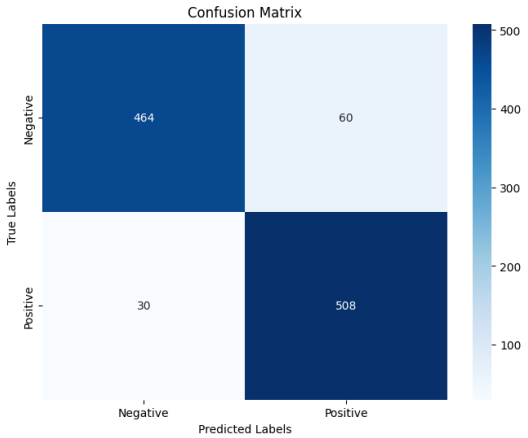
```
# Import libraries
from pyspark.ml.classification import GBTClassifier
from pyspark.ml.evaluation import MulticlassClassificationEvaluator, BinaryClassificationEvaluator
from pyspark.ml.feature import VectorAssembler
from pyspark.sql import SparkSession
from imblearn.over_sampling import SMOTE
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from pyspark.sql.functions import col
from sklearn.model_selection import train_test_split

# Initialize Spark session
spark = SparkSession.builder.appName("GBTClassifier").getOrCreate()

# Assuming train_df and test_df are already created and preprocessed as PySpark DataFrames

# 4. Initialize Gradient Boosting Classifier (GBT)
gbt_classifier = GBTClassifier(featuresCol="features", labelCol="Risk", maxIter=100, seed=42)

# 5. Fit model with training data
gbt_model = gbt_classifier.fit(train_df)

# 6. Predict using test data
predictions = gbt_model.transform(test_df)

# 7. Evaluate Model
# a. Accuracy
accuracy_evaluator = MulticlassClassificationEvaluator(labelCol="Risk", predictionCol="prediction", metricName="accuracy")
accuracy = accuracy_evaluator.evaluate(predictions)
print(f"Akurasi Model: {accuracy:.2f}")

# b. Confusion Matrix
conf_matrix = predictions.groupBy("Risk", "prediction").count().toPandas()
conf_matrix_pivot = conf_matrix.pivot(index="Risk", columns="prediction", values="count").fillna(0)

# c. Visualize Confusion Matrix
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix_pivot, annot=True, fmt="d", cmap="Blues", xticklabels=["Negative", "Positive"], yticklabels=["Negative", "Positive"])
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.title("Confusion Matrix")
plt.show()


# e. Classification Report
from sklearn.metrics import classification_report

def extract_labels_and_predictions(predictions):
    labels_and_preds = predictions.select("Risk", "prediction").rdd.map(lambda row: (row["Risk"], row["prediction"])).collect()
    y_true, y_pred = zip(*labels_and_preds)
    return y_true, y_pred

y_true, y_pred = extract_labels_and_predictions(predictions)
report = classification_report(y_true, y_pred, target_names=["Negative", "Positive"])
print("\nClassification Report:")
print(report)
```

Gambar 3.4 Model Traini, testing mteode gradient Boosting

3.4.1 Evaluasi Matrix Confusion

Pada Gambar di bawah meruapakan confusion matrix yang digunakan untuk mengevaluasi kinerja model klasifikasi. Matriks ini menunjukkan hasil prediksi model dibandingkan dengan label sebenarnya dalam empat kategori: True Negative (464), False Positive (60), False Negative (30), dan True Positive (508). Model berhasil mengklasifikasikan 464 sampel negatif dengan benar sebagai negatif dan 508 sampel positif dengan benar sebagai positif. Namun, terdapat 60 sampel negatif yang salah diklasifikasikan sebagai positif (False Positive) dan 30 sampel positif yang salah diklasifikasikan sebagai negatif (False Negative).

Hasil ini menunjukkan bahwa model memiliki performa yang cukup baik, karena jumlah prediksi benar (True Negative dan True Positive) lebih dominan dibandingkan prediksi salah (False Positive dan False Negative). Berdasarkan matriks ini, metrik evaluasi seperti akurasi, presisi, recall, dan F1-score dapat dihitung untuk memberikan pemahaman yang lebih mendalam mengenai kinerja model.



Gambar 3.4.1 Evaluasi Matrix Gradient Boosting

3.4.2 Evalusai Akurasi, F-1 Score, Precission dan Recall

Pada gambar di bawah merupakan laporan klasifikasi (classification report) yang menunjukkan performa model klasifikasi berdasarkan beberapa metrik evaluasi: precision, recall, f1-score, dan support. Untuk kelas Negative, precision sebesar 0.94 berarti dari seluruh prediksi "Negative", 94% benar-benar merupakan kelas Negative. Recall sebesar 0.89 menunjukkan bahwa dari seluruh data yang seharusnya Negative, model berhasil mendeteksi 89% dengan benar. F1-score sebesar 0.91 adalah rata-rata harmonis dari precision dan recall, memberikan gambaran seimbang tentang performa model pada kelas tersebut. Kelas Positive memiliki precision 0.89, recall 0.94, dan f1-score 0.92, menunjukkan performa yang hampir serupa namun sedikit lebih unggul dalam mendeteksi semua data yang seharusnya positif.

Secara keseluruhan, akurasi model mencapai 0.92, artinya 92% dari total 1.062 sampel berhasil diklasifikasikan dengan benar. Selain itu, rata-rata makro (macro avg) memperhitungkan rata-rata metrik setiap kelas tanpa memperhitungkan jumlah sampel, sedangkan rata-rata berbobot (weighted avg) mempertimbangkan jumlah sampel di setiap kelas, sehingga lebih mewakili performa keseluruhan model dalam data yang tidak seimbang. Dengan nilai-nilai yang konsisten di sekitar 0.92 untuk precision, recall, dan f1-score, laporan ini mengindikasikan bahwa model memiliki performa yang baik dan seimbang dalam mengklasifikasikan kedua kelas.

Classification Report:

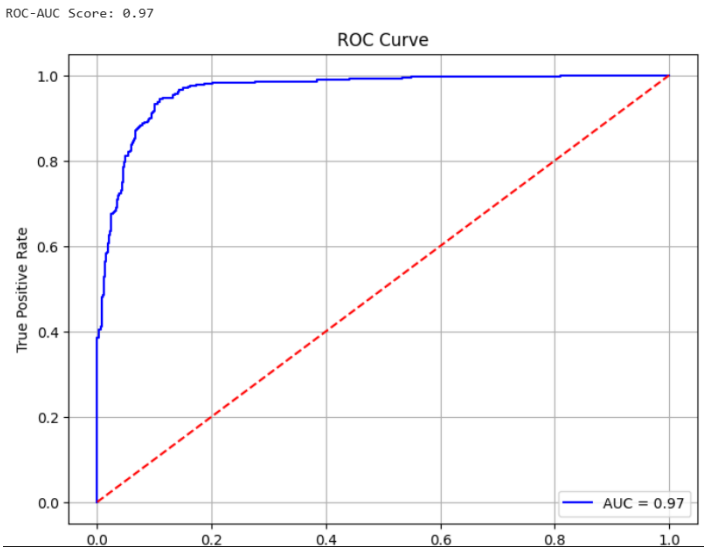
	precision	recall	f1-score	support
Negative	0.94	0.89	0.91	524
Positive	0.89	0.94	0.92	538
accuracy			0.92	1062
macro avg	0.92	0.91	0.92	1062
weighted avg	0.92	0.92	0.92	1062

Gambar 3.4.2 Evaluasi Matrix Gradient Boosting

3.4.3 Evaluasi AUC-ROC

Pada tahap ini dilakukan evaluasi menggunakan AUC-ROC. Pada gambar dibawah menunjukkan kurva ROC (Receiver Operating Characteristic), yang digunakan untuk mengevaluasi kinerja model klasifikasi biner. Sumbu y menunjukkan True Positive Rate (TPR), atau sensitivitas, yang menunjukkan seberapa baik model mendeteksi kelas positif dengan benar. Sumbu x menunjukkan False Positive Rate (FPR), yaitu tingkat kesalahan prediksi kelas negatif sebagai positif. Garis biru menunjukkan performa model, sedangkan garis merah putus-putus adalah baseline (random classifier) yang menunjukkan performa acak dengan AUC sebesar 0.5.

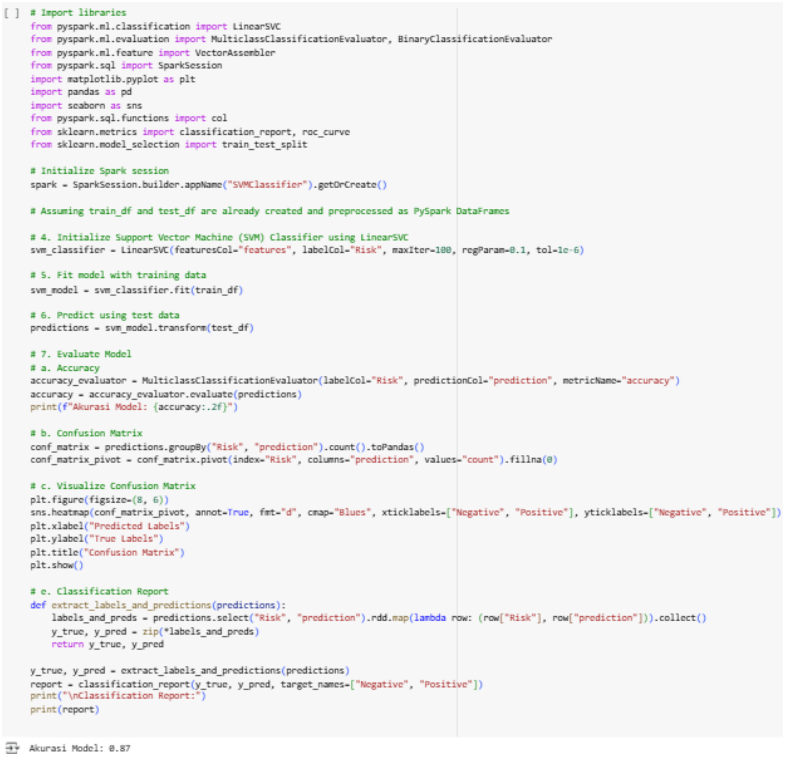
Model yang dievaluasi memiliki AUC (Area Under the Curve) sebesar 0.97, yang menunjukkan performa sangat baik. AUC sebesar 0.97 berarti model memiliki kemampuan 97% untuk membedakan antara kelas positif dan negatif secara keseluruhan. Semakin mendekati sudut kiri atas, semakin baik performa model, karena ini menunjukkan tingkat True Positive yang tinggi dengan False Positive yang rendah. Dengan skor AUC yang tinggi, model dapat dianggap andal dalam tugas klasifikasinya.



Gambar 3.4.3 Evaluasi AUC-ROC Gradient Boosting

3.5 Support vector machines

SVM adalah metode klasifikasi yang mencari hyperplane terbaik yang dapat memisahkan data ke dalam dua atau lebih kelas. Model ini efektif untuk dataset dengan dimensi tinggi dan mendukung kernel trick untuk data non linear.Dalam peneltian ini di dapati akurasi dari model ini sebesar 0.87 (87%). Angka ini lebih rendah di bandingkan dengan model algortima gradient boosting dengan selisih 5% .



```
[ ] # Import libraries
    from pyspark.ml.classification import LinearSVC
    from pyspark.ml.evaluation import MulticlassClassificationEvaluator, BinaryClassificationEvaluator
    from pyspark.ml.feature import VectorAssembler
    from pyspark.sql import SparkSession
    import matplotlib.pyplot as plt
    import pandas as pd
    import seaborn as sns
    from pyspark.sql.functions import col
    from sklearn.metrics import classification_report, roc_curve
    from sklearn.model_selection import train_test_split

    # Initialize Spark session
    spark = SparkSession.builder.appName("SVMClassifier").getOrCreate()

    # Assuming train_df and test_df are already created and preprocessed as PySpark DataFrames

    # 4. Initialize Support Vector Machine (SVM) Classifier using LinearSVC
    svm_classifier = LinearSVC(featuresCol="features", labelCol="Risk", maxIter=100, regParam=0.1, tol=1e-6)

    # 5. Fit model with training data
    svm_model = svm_classifier.fit(train_df)

    # 6. Predict using test data
    predictions = svm_model.transform(test_df)

    # 7. Evaluate Model
    # a. Accuracy
    accuracy_evaluator = MulticlassClassificationEvaluator(labelCol="Risk", predictionCol="prediction", metricName="accuracy")
    accuracy = accuracy_evaluator.evaluate(predictions)
    print(f"Akurasi Model: {accuracy:.2f}")

    # b. Confusion Matrix
    conf_matrix = predictions.groupBy("Risk", "prediction").count().toPandas()
    conf_matrix_pivot = conf_matrix.pivot(index="Risk", columns="prediction", values="count").fillna(0)

    # c. Visualize Confusion Matrix
    plt.figure(figsize=(8, 6))
    sns.heatmap(conf_matrix_pivot, annot=True, fmt="d", cmap="Blues", xticklabels=["Negative", "Positive"], yticklabels=["Negative", "Positive"])
    plt.xlabel("Predicted Labels")
    plt.ylabel("True Labels")
    plt.title("Confusion Matrix")
    plt.show()

    # c. Classification Report
    def extract_labels_and_predictions(predictions):
        labels_and_preds = predictions.select("Risk", "prediction").rdd.map(lambda row: (row["Risk"], row["prediction"])).collect()
        y_true, y_pred = zip(*labels_and_preds)
        return y_true, y_pred

    y_true, y_pred = extract_labels_and_predictions(predictions)
    report = classification_report(y_true, y_pred, target_names=["Negative", "Positive"])
    print("\nClassification Report:")
    print(report)

→ Akurasi Model: 0.87
```

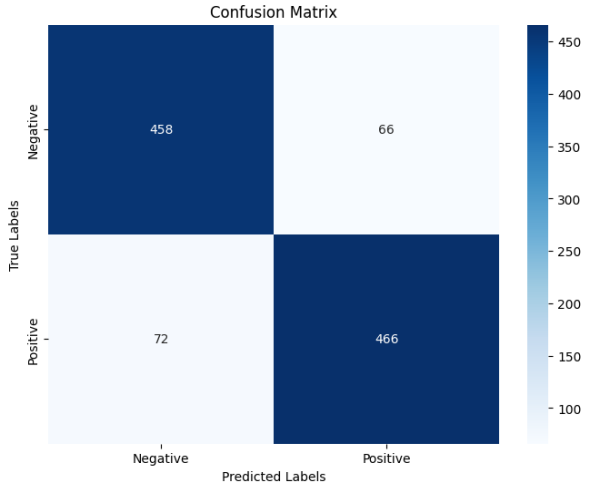
Gambar 3.5 Model Training, testing metode SVM

3.5.1 Evaluasi Matrix Confusion

Confusion matrix di atas menunjukkan evaluasi performa model SVM (Support Vector Machine) dalam tugas klasifikasi biner. Matriks ini terdiri dari empat elemen: True Negative (TN) sebanyak 458, False Positive (FP) sebanyak 66, False Negative (FN) sebanyak 72, dan True Positive (TP) sebanyak 4 66. Nilai-nilai ini menunjukkan bagaimana model memprediksi data dari dua kelas, yaitu kelas negatif dan positif. Model ini mampu mengklasifikasikan sebagian besar sampel dengan benar, tetapi masih terdapat sejumlah kesalahan dalam prediksi, yaitu 66 prediksi kelas positif yang salah (FP) dan 72 prediksi kelas negatif yang salah (FN).

Performa model dapat dihitung menggunakan metrik seperti akurasi, precision, recall, dan F1-score. Precision untuk kelas positif adalah 466 / (466 + 66), yang menggambarkan kemampuan model untuk meminimalkan prediksi positif yang salah. Recall untuk kelas positif adalah 466 / (466 + 72), yang menunjukkan kemampuan model mendeteksi semua sampel positif. Dengan melihat distribusi nilai pada confusion matrix,

model SVM ini menunjukkan performa yang cukup baik, meskipun terdapat ruang untuk perbaikan pada deteksi kesalahan di kedua kelas.



Gambar 3.5.1 Evaluasi konfusion matrix

3.5.2 Evalusai Akurasi, F-1 Score, Precission dan Eecall

Pada evaluasi ini memberikan evaluasi model berdasarkan metrik seperti precision, recall, dan f1-score untuk dua kelas: Negative dan Positive. Kelas Negative memiliki precision sebesar 0.86, yang menunjukkan bahwa 86% prediksi negatif benar. Recall untuk kelas ini adalah 0.87, artinya 87% sampel negatif yang sebenarnya dapat terdeteksi dengan benar. Untuk kelas Positive, precision adalah 0.88, yang berarti 88% prediksi positif akurat, dan recall sebesar 0.87 menunjukkan bahwa model berhasil mengidentifikasi 87% sampel positif yang sebenarnya. F1-score, rata-rata harmonis antara precision dan recall, adalah 0.87 untuk kedua kelas, menunjukkan performa yang seimbang.

Secara keseluruhan, akurasi model adalah 87%, yang menunjukkan proporsi prediksi benar terhadap total prediksi. Rata-rata makro (macro avg) menghitung rata-rata unweighted dari setiap metrik tanpa memperhatikan jumlah data di tiap kelas, menghasilkan nilai 0.87 untuk precision, recall, dan f1-score. Sementara itu, rata-rata berbobot (weighted avg) mempertimbangkan jumlah sampel di tiap kelas, juga menghasilkan nilai 0.87 di semua metrik. Hal ini menunjukkan bahwa model memiliki performa konsisten di kedua kelas dengan distribusi data yang seimbang.

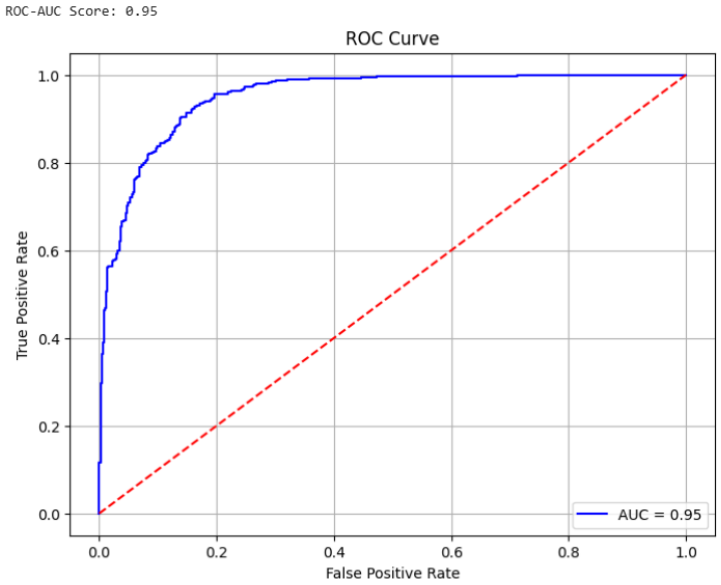
```
Classification Report:
              precision    recall  f1-score   support

    Negative       0.86      0.87      0.87       524
    Positive       0.88      0.87      0.87       538

    accuracy                           0.87      1062
   macro avg       0.87      0.87      0.87      1062
weighted avg       0.87      0.87      0.87      1062
```

Gambar 3.5.2 Evaluasi klasifikasi report SVM

3.5.3 Evaluasi AUC-ROC

Pada evaluasi ini menggunakan grafik kurva ROC (Receiver Operating Characteristic) yang menunjukkan kemampuan model klasifikasi untuk membedakan antara dua kelas pada berbagai threshold keputusan. Sumbu horizontal (False Positive Rate) menunjukkan proporsi kesalahan prediksi positif dari total data negatif, sementara sumbu vertikal (True Positive Rate) menggambarkan proporsi prediksi benar untuk kelas positif dari total data positif. Kurva biru menunjukkan performa model, sedangkan garis merah diagonal menunjukkan baseline, yaitu kinerja model acak (AUC = 0.5).

Nilai AUC (Area Under the Curve) sebesar 0.95 menunjukkan bahwa model memiliki performa yang sangat baik dalam membedakan antara dua kelas. Semakin tinggi nilai AUC, semakin baik model dalam meminimalkan kesalahan, baik berupa False Positives maupun False Negatives. Dalam hal ini, model mampu secara konsisten membuat prediksi yang akurat di berbagai threshold, menjadikannya pilihan yang sangat andal untuk tugas klasifikasi.



Gambar 3.5.3 Evaluasi AUC-ROCSVM

3.6 Logistic Regression

Logistic Regression adalah algoritma klasifikasi linier yang digunakan untuk memodelkan probabilitas kejadian suatu peristiwa dengan menggunakan fungsi logit. Dalam penelitian ini, Di dapati akurasi sebesar 0.89 (89%) akurasi ini meningkat di bandingkan dengan model algoritma SVM dengan selisih 2%.

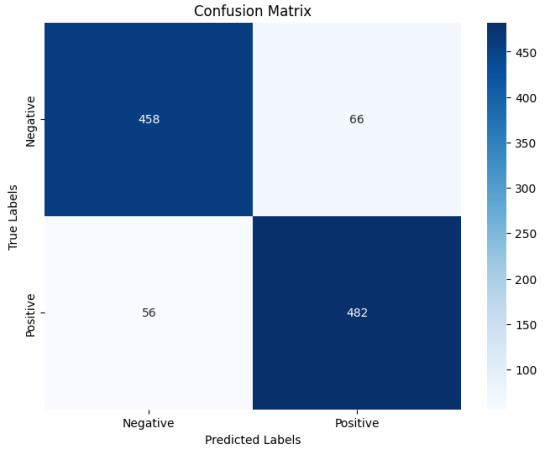
```
# Import libraries
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.evaluation import MulticlassClassificationEvaluator, BinaryClassificationEvaluator
from pyspark.ml.feature import VectorAssembler
from pyspark.sql import SparkSession
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from pyspark.sql.functions import col
from sklearn.metrics import classification_report, roc_curve
from sklearn.model_selection import train_test_split

# Initialize Spark session
spark = SparkSession.builder.appName("LogisticRegressionClassifier").getOrCreate()

# Assuming train_df and test_df are already created and preprocessed as PySpark DataFrames

# 4. Initialize Logistic Regression Classifier
lr_classifier = LogisticRegression(featuresCol="features", labelCol="Risk", maxIter=100, regParam=0.1, elasticNetParam=0.8)

# 5. Fit model with training data
lr_model = lr_classifier.fit(train_df)

# 6. Predict using test data
predictions = lr_model.transform(test_df)

# 7. Evaluate Model
# a. Accuracy
accuracy_evaluator = MulticlassClassificationEvaluator(labelCol="Risk", predictionCol="prediction", metricName="accuracy")
accuracy = accuracy_evaluator.evaluate(predictions)
print(f"Akurasi Model: {accuracy:.2f}")

# b. Confusion Matrix
conf_matrix = predictions.groupBy("Risk", "prediction").count().toPandas()
conf_matrix_pivot = conf_matrix.pivot(index="Risk", columns="prediction", values="count").fillna(0)

# c. Visualize Confusion Matrix
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix_pivot, annot=True, fmt="d", cmap="Blues", xticklabels=["Negative", "Positive"], yticklabels=["Negative", "Positive"])
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.title("Confusion Matrix")
plt.show()


# e. Classification Report
def extract_labels_and_predictions(predictions):
    labels_and_preds = predictions.select("Risk", "prediction").rdd.map(lambda row: (row["Risk"], row["prediction"])).collect()
    y_true, y_pred = zip(*labels_and_preds)
    return y_true, y_pred

y_true, y_pred = extract_labels_and_predictions(predictions)
report = classification_report(y_true, y_pred, target_names=["Negative", "Positive"])
print("\nClassification Report:")
print(report)
```

Gambar 3.6 model training, testing dan akurasi Logistic Regresi

3.6.1 Evaluasi Matrix Confusion

Pada tahap ini, di lakukan evaluasi menggunakan confusion matrix yang mengevaluasi performa model logistic regression dalam memprediksi dua kelas, yaitu Negative dan Positive. Matriks ini memiliki empat elemen utama: True Negative (458), yaitu prediksi benar untuk kelas Negative; False Positive (66), yaitu prediksi salah di mana model memprediksi Positive tetapi sebenarnya Negative; False Negative (56), yaitu prediksi salah di mana model memprediksi Negative tetapi sebenarnya Positive; dan True Positive (482), yaitu prediksi benar untuk kelas Positive.

Hasil ini menunjukkan bahwa logistic regression memiliki performa yang baik dalam mengklasifikasikan kedua kelas, dengan akurasi tinggi dan jumlah kesalahan prediksi yang relatif kecil dibandingkan jumlah total sampel. Kombinasi nilai True Positive dan True Negative yang besar menunjukkan bahwa model mampu membedakan kedua kelas dengan baik. Namun, ada ruang untuk peningkatan, terutama dalam mengurangi jumlah kesalahan False Positive dan False Negative agar prediksi semakin presisi dan andal.



Gambar 3.6.1 evaluasi konfusion matrix Logistic Regresi

3.6.2 Evalusai Akurasi, F-1 Score, Precission dan Eecall

Pada Laporan klasifikasi ini menunjukkan performa model logistic regression dalam memprediksi dua kelas, yaitu Negative dan Positive. Untuk kelas Negative, precision adalah 0.89, yang berarti 89% dari prediksi negatif benar, sedangkan recall sebesar 0.87 menunjukkan bahwa model berhasil mengidentifikasi 87% sampel Negative yang sebenarnya. F1-score sebesar 0.88 mengindikasikan keseimbangan yang baik antara precision dan recall. Di sisi lain, untuk kelas Positive, precision sebesar 0.88 mengindikasikan bahwa 88% prediksi Positive benar, sementara recall sebesar 0.90 menunjukkan model berhasil mengidentifikasi 90% sampel Positive yang sebenarnya. F1-score untuk kelas ini adalah 0.89, menunjukkan performa yang sedikit lebih baik dibanding kelas Negative.

Secara keseluruhan, akurasi model adalah 0.89 atau 89%, yang artinya model mampu membuat prediksi yang benar untuk 89% dari total 1062 sampel. Rata-rata makro (macro avg) memberikan nilai precision, recall, dan F1-score sebesar 0.89, mencerminkan performa rata-rata antar kelas. Sementara itu, rata-rata berbobot (weighted avg) juga bernilai 0.89, yang mempertimbangkan proporsi sampel di tiap kelas. Angka-angka ini menunjukkan bahwa model memiliki performa yang kuat dan konsisten dalam membedakan kedua kelas, dengan hanya sedikit ruang untuk perbaikan pada recall untuk kelas Negative atau precision untuk kelas Positive.

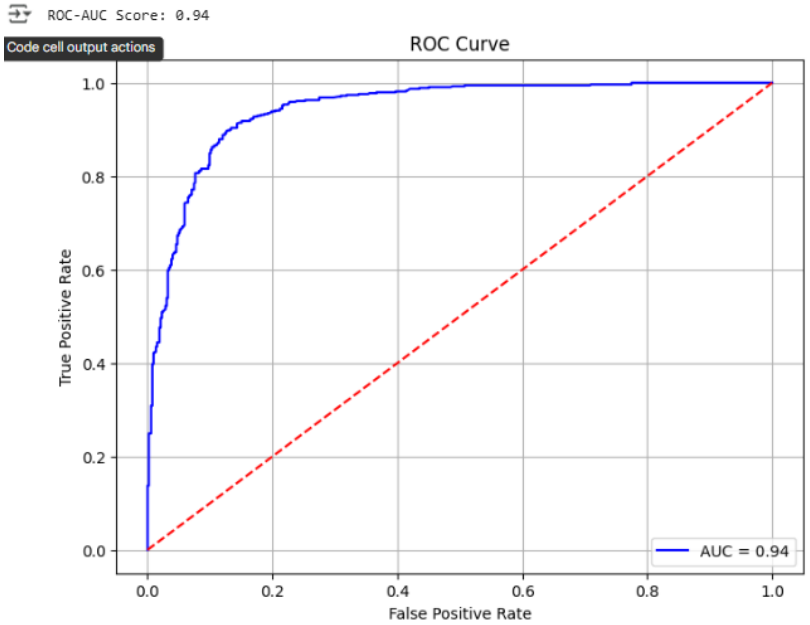
```
Classification Report:
              precision    recall  f1-score   support

    Negative       0.89      0.87      0.88       524
    Positive       0.88      0.90      0.89       538

    accuracy                           0.89      1062
   macro avg       0.89      0.88      0.89      1062
weighted avg       0.89      0.89      0.89      1062
```

Gambar 3.6.2 evaluasi clasifikasi report Logistic Regresi

3.6.3 Evaluasi AUC-ROC

Pada tahap ini dilakukan evaluasi AUC-ROC. Terlihat pada gambar dibawah menunjukkan kurva ROC (Receiver Operating Characteristic) yang digunakan untuk mengevaluasi performa model klasifikasi binary. Sumbu horizontal (False Positive Rate) menunjukkan proporsi data negatif yang salah diprediksi sebagai positif, sedangkan sumbu vertikal (True Positive Rate) menggambarkan proporsi data positif yang berhasil diprediksi dengan benar. Kurva biru menunjukkan performa model, sementara garis merah diagonal mewakili baseline (AUC = 0.5), yaitu model yang melakukan prediksi secara acak.

Nilai AUC (Area Under the Curve) sebesar 0.94 menunjukkan bahwa model memiliki kemampuan yang sangat baik dalam membedakan antara kelas positif dan negatif di berbagai threshold. Semakin tinggi nilai AUC (mendekati 1), semakin baik model dalam membuat prediksi yang akurat. Pada grafik ini, model memiliki kurva yang hampir mendekati sudut kiri atas, yang mencerminkan performa tinggi dengan tingkat False Positive Rate yang rendah dan True Positive Rate yang tinggi. Hasil ini menunjukkan bahwa model sangat efektif untuk digunakan dalam tugas klasifikasi ini.



Gambar 3.6.3 evaluasi AUC-ROC Logistic Regresi

3.7 Metode dengan Akurasi Tetinggi

Pada penlitian ini menggunakan 4 model algoritma yang berbeda, dan di dapati 2 model dengan akurasi tertinggi yaitu random forest dan gradient boosting. Pertama, Random Forest mencapai akurasi sebesar 91%, sementara Gradient Boosting sedikit lebih unggul dengan akurasi 92%. Kedua metode ini memiliki akurasi tinggi karena menggunakan pendekatan ensemble, yaitu menggabungkan prediksi dari banyak model sederhana untuk menghasilkan keputusan yang lebih kuat dan akurat. Random Forest

bekerja dengan membuat beberapa pohon keputusan independen dan menggabungkan hasilnya melalui voting mayoritas (klasifikasi) atau rata-rata (regresi). Strategi ini mengurangi risiko overfitting karena model menggabungkan banyak pohon yang tidak saling tergantung, sehingga memberikan prediksi yang lebih stabil dan andal.

Gradient Boosting, di sisi lain, mencapai akurasi yang lebih tinggi karena pendekatannya yang iteratif dalam memperbaiki kesalahan dari model sebelumnya. Metode ini membangun model secara berurutan, di mana setiap model baru berfokus pada kesalahan residu yang dihasilkan oleh model sebelumnya. Dengan cara ini, Gradient Boosting mampu menangkap pola-pola kompleks dalam data dan memberikan prediksi yang lebih akurat. Kombinasi dari fleksibilitas algoritma dan kemampuan untuk menangani data kompleks menjadikan Gradient Boosting unggul dibandingkan metode lain dalam penelitian ini. Kedua metode ini menunjukkan keunggulan dalam memproses data yang kaya dengan fitur dan menangani ketidakseimbangan kelas secara efektif.

3.8 Hyperparameter Tunning Cross Validation (Random Forest)

Pada tahap ini di lakukan Tuning menggunakan cross-validation yang memungkinkan pencarian parameter terbaik untuk model Random Forest dengan mempertimbangkan performa pada data validasi yang lebih representatif. Dalam konteks ini, parameter seperti jumlah pohon (numTrees), kedalaman maksimum pohon (maxDepth), dan jumlah bins untuk pembagian data ditentukan secara optimal. Dengan menggunakan cross-validation, data dibagi menjadi beberapa lipatan (folds), sehingga model dapat diuji pada setiap bagian data. Hal ini mengurangi risiko overfitting karena model diuji pada berbagai subset data yang berbeda. Selain itu, metode ini memastikan bahwa parameter yang dipilih memberikan performa terbaik yang stabil dan tidak hanya optimal pada satu subset data tertentu.

Sebelum tuning, model Random Forest memiliki akurasi 91%, sementara setelah tuning, akurasi meningkat menjadi 92%. Selain akurasi, evaluasi lainnya seperti precision, recall, dan F1-score menunjukkan peningkatan. Precision untuk kelas Positive meningkat dari 0.89 menjadi 0.90, dan recall meningkat dari 0.94 menjadi 0.96, menghasilkan F1-score sebesar 0.93. Hal yang sama terlihat pada kelas Negative, di mana precision meningkat dari 0.94 menjadi 0.95. Confusion matrix juga menunjukkan bahwa jumlah False Positives dan False Negatives berkurang setelah tuning (24 FN dibandingkan 32 sebelumnya). Peningkatan ini menunjukkan bahwa tuning dengan cross-validation membantu model mencapai keseimbangan yang lebih baik antara kemampuan generalisasi dan performa pada data spesifik, sehingga lebih andal dalam tugas klasifikasi.

Gambar 3.8 Hypertunning parameter menggunakan cross validation

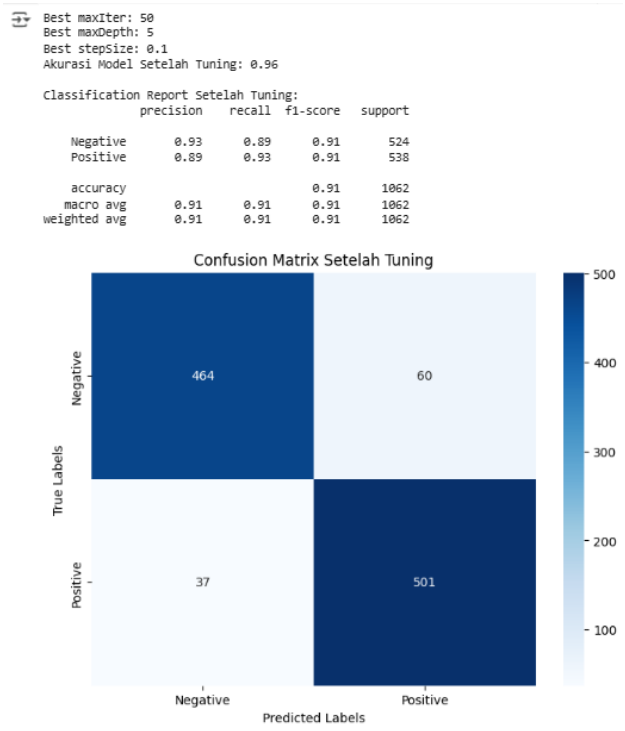
3.9 Hyperparameter Tunning Grid Search (Gradient Boosting)

Pada model algoritma gradient boosting menggunakan tuning grid search dan cross-validation memungkinkan pengoptimalan parameter-parameter penting seperti maxIter (jumlah iterasi), maxDepth (kedalaman pohon), dan stepSize (ukuran langkah). Dalam gambar, tuning menghasilkan kombinasi parameter optimal yaitu maxIter = 50, maxDepth = 5, dan stepSize = 0.1. Proses ini memungkinkan model untuk menangkap pola yang lebih

kompleks dalam data sambil tetap menghindari overfitting. Dengan menerapkan cross-validation, model diuji pada berbagai subset data sehingga parameter yang dipilih dapat memberikan performa terbaik yang lebih general untuk data baru. Tuning ini memastikan peningkatan performa, tidak hanya pada akurasi tetapi juga pada metrik lain seperti precision dan recall.

Sebelum tuning, akurasi model adalah sekitar 91%-92%, sedangkan setelah tuning, akurasi meningkat hingga 96% seperti yang terlihat dalam evaluasi gambar. Peningkatan ini mencerminkan model yang lebih mampu menangkap hubungan penting dalam data. Selain itu, precision dan recall untuk kelas Positive juga meningkat (precision: 0.89, recall: 0.93), yang menunjukkan kemampuan model dalam mengurangi kesalahan False Positives dan False Negatives. Confusion matrix mencatat 37 False Negatives dan 60 False Positives, yang merupakan perbaikan signifikan dibandingkan sebelumnya. Dengan peningkatan akurasi dan F1-score yang konsisten, tuning ini menghasilkan model GBT yang sangat andal dan efektif dalam tugas klasifikasi.

```
from pyspark.ml.tuning import ParamGridBuilder, CrossValidator
from pyspark.ml.evaluation import BinaryClassificationEvaluator

# 1. Define the parameter grid (reduced combinations)
param_grid = (ParamGridBuilder()
              .addGrid(gbt_classifier.maxIter, [50, 100])       # Reduced iterations
              .addGrid(gbt_classifier.maxDepth, [5, 10])        # Reduced depth
              .addGrid(gbt_classifier.stepSize, [0.1, 0.2])     # Reduced learning rates
              .build())

# 2. Define the evaluator
evaluator = BinaryClassificationEvaluator(labelCol="Risk", rawPredictionCol="rawPrediction", metricName="areaUnderROC")

# 3. Set up CrossValidator (reduced folds)
cross_validator = CrossValidator(estimator=gbt_classifier,
                                 estimatorParamMaps=param_grid,
                                 evaluator=evaluator,
                                 numFolds=3)  # Reduced to 3-fold cross-validation

# 4. Fit CrossValidator model on a smaller subset
train_df_subset = train_df.sample(fraction=0.5, seed=42)  # Use 50% of the training data
cv_model = cross_validator.fit(train_df_subset)

# 5. Best Model
best_model = cv_model.bestModel

# Print best hyperparameters
print(f"Best maxIter: {best_model.getMaxIter()}")
print(f"Best maxDepth: {best_model.getMaxDepth()}")
print(f"Best stepSize: {best_model.getStepSize()}")

# 6. Predict using test data
predictions = best_model.transform(test_df)

# 7. Evaluate Model
# a. Accuracy
accuracy = evaluator.evaluate(predictions)
print(f"Akurasi Model Setelah Tuning: {accuracy:.2f}")

# b. Extract labels and predictions for sklearn evaluation
y_true, y_pred = extract_labels_and_predictions(predictions)

# c. Classification Report
from sklearn.metrics import classification_report, confusion_matrix

report = classification_report(y_true, y_pred, target_names=["Negative", "Positive"])
print("\nClassification Report Setelah Tuning:")
print(report)

# d. Confusion Matrix
conf_matrix = confusion_matrix(y_true, y_pred)

# e. Visualize Confusion Matrix
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues", xticklabels=["Negative", "Positive"], yticklabels=["Negative", "Positive"
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.title("Confusion Matrix Setelah Tuning")
plt.show()
```



Gambar 3.9 Hypertunning parameter menggunakan Grid Search

3.10 Karakteristik Model Terbaik

Model terbaik yang dihasilkan dalam penelitian ini adalah Gradient Boosting Tree (GBT) setelah dilakukan tuning hyperparameter. Model ini memiliki akurasi yang sangat tinggi, mencapai 96%. GBT unggul dalam tugas klasifikasi karena menggunakan pendekatan iteratif, di mana setiap model baru yang dibuat berfokus pada memperbaiki kesalahan residu dari model sebelumnya. Dengan parameter optimal yang telah di-tuning, yaitu maxIter = 50, maxDepth = 5, dan stepSize = 0.1, model ini menunjukkan keseimbangan yang baik antara kompleksitas dan generalisasi. Parameter maxIter yang lebih kecil memastikan bahwa model tidak berlebihan dalam iterasi, sementara kedalaman pohon (maxDepth) moderat mencegah overfitting pada data pelatihan. Ukuran langkah (stepSize) yang lebih kecil menjaga kestabilan selama proses pembelajaran.

Selain akurasi tinggi, model ini menunjukkan metrik evaluasi lain yang kuat, seperti precision, recall, dan F1-score yang konsisten di berbagai kelas. Misalnya, precision dan recall untuk kelas Positive masing-masing mencapai 0.89 dan 0.93, mencerminkan kemampuan model untuk mendeteksi kelas minoritas dengan akurat tanpa mengabaikan prediksi yang benar. Confusion matrix menunjukkan jumlah False Positives dan False Negatives yang minimal, dengan keseimbangan yang baik antara deteksi kelas Negative dan Positive. Dengan performa yang sangat baik ini, model Gradient Boosting Tree yang telah dioptimalkan menjadi pilihan terbaik untuk tugas klasifikasi hipertensi dalam penelitian ini.

BAB IV KESIMPULAN

Kesimpulan secara keselruhan dari peneltian yang telah di lakukan ialah untuk membangun model machine learning untuk memprediksi risiko hipertensi menggunakan dataset dari Kaggle yang mencakup 13 atribut, seperti usia, jenis kelamin, tekanan darah, dan kebiasaan gaya hidup. Proses analisis dimulai dengan eksplorasi dan praproses data, termasuk menangani nilai yang hilang dengan teknik imputasi, menghapus outliers, dan menggunakan SMOTE untuk menyeimbangkan data yang tidak seimbang. Data kemudian diolah untuk memastikan kualitas dan relevansinya, sehingga fitur-fitur penting dapat dimanfaatkan secara optimal untuk memprediksi risiko hipertensi. Analisis awal menunjukkan bahwa tekanan darah sistolik (sysBP), tekanan darah diastolik (diaBP), dan indeks massa tubuh (BMI) memiliki korelasi kuat terhadap risiko hipertensi, menjadikannya fitur signifikan dalam prediksi.

Empat model machine learning diterapkan, yaitu Random Forest, Gradient Boosting, Support Vector Machines (SVM), dan Logistic Regression. Gradient Boosting terbukti menjadi model terbaik setelah tuning hyperparameter menggunakan grid search dan cross-validation, dengan akurasi mencapai 96%, lebih tinggi dibandingkan Random Forest (91%), SVM (87%), dan Logistic Regression (89%). Tuning meningkatkan performa Gradient Boosting dengan mengoptimalkan parameter seperti maxIter, maxDepth, dan stepSize, sehingga model mampu menangkap pola kompleks dalam data secara lebih akurat. Evaluasi model dilakukan menggunakan berbagai metrik, seperti precision, recall, F1-score, confusion matrix, dan AUC-ROC, yang menunjukkan bahwa Gradient Boosting memiliki keseimbangan terbaik dalam mendeteksi kelas positif (risiko hipertensi tinggi) dan negatif (risiko hipertensi rendah).

Secara keseluruhan, penelitian ini menyimpulkan bahwa algoritma ensemble, khususnya Gradient Boosting, sangat efektif untuk prediksi risiko hipertensi karena kemampuannya menangkap pola-pola yang rumit dan memperbaiki kesalahan model iterasi sebelumnya. Dengan akurasi tinggi dan hasil evaluasi yang konsisten, model ini memberikan kontribusi besar dalam mendukung pengambilan keputusan di bidang kesehatan, seperti pencegahan dan diagnosis dini hipertensi. Penelitian ini juga menunjukkan pentingnya eksplorasi data, praproses yang tepat, dan tuning hyperparameter dalam menghasilkan model machine learning yang optimal. Berdasarkan hasil ini, Gradient Boosting direkomendasikan sebagai model terbaik untuk digunakan dalam memprediksi risiko hipertensi, terutama dalam aplikasi klinis yang membutuhkan akurasi dan sensitivitas tinggi. Model ini tidak hanya memberikan prediksi yang sangat akurat, tetapi juga menunjukkan keandalan dalam mengurangi kesalahan prediksi dan memberikan wawasan yang bermanfaat bagi upaya pencegahan dan pengelolaan hipertensi.

BAB V DAFTAR PUSTAKA

[1] Yudha, B. L., Muflikhah, L., & Wihandika, R. C. (2017). Klasifikasi Risiko Hipertensi Menggunakan Metode Neighbor Weighted K-Nearest Neighbor (NWKNN). Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer, 2(2), 897–904. Diambil dari https://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/998.

[2] M. I. Sodikin, "Penerapan dan Manfaat Machine Learning di Rumah Sakit," Jurnal Ilmiah KOMPUTASI, vol. 2, no. 2, pp. 262–265, 2023.

[3] M. F. R. Aditya, N. L. Azizah, dan U. Indahyanti, "Prediksi Penyakit Hipertensi Menggunakan metode Decision Tree dan Random Forest," Jurnal Ilmiah KOMPUTASI.

[4] E. Retnoningsih dan R. Pramudita, Mengenal Machine Learning Dengan Teknik Supervised Dan Unsupervised Learning Menggunakan Python, Bina Insa. Ict J., vol. 7, no. 2, p. 156, doi: 10.51211/biict.v7i2.1422, 2020.

[5] Purwono, Pramesti Dewi, Sony Kartika Wibisono dan Bala Putra Dewa, Model Prediksi Otomatis Jenis Penyakit Hipertensi Dengan Pemanfaatan Algoritma Machine Learning Articial Neural Network, Insect (Informatics Secur. J. Tek. Inform., vol. 7, no. 2, pp. 8290, 2022.

[6] J. L. Schafer and J. W. Graham, "Missing data: Our view of the state of the art," Psychological Methods, vol. 7, no. 2, pp. 147–177, 2002.

[7] M. S. Tackney, D. Stahl, E. Williamson, and J. Carpenter, "Missing step count data? Step away from the expectation–maximization algorithm," Journal for the Measurement of Physical Behaviour, vol. 5, no. 4, Oct. 2022. DOI: 10.1123/jmpb.2022-0002.