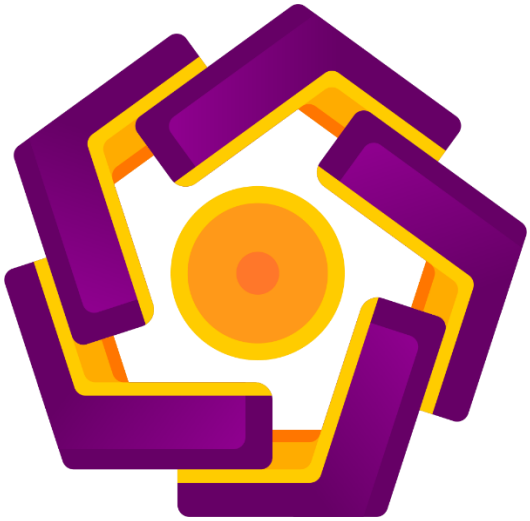
UJIAN AKHIR SEMESTER

BIG DATA & PREDICTIVE ANALYTICS

Dosen: Anna Baita, M.Kom



Disusun oleh

Habib Baitul Hamdi	22.11.4571
Zahran Nugraha	22.11.4598

PROGRAM STUDI INFORMATIKA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS AMIKOM YOGYAKARTA

2025

A. Latar Belakan

Tesla adalah salah satu perusahaan terkemuka di dunia dalam industri otomotif yang dikenal terutama karena inovasinya dalam kendaraan listrik. Didirikan pada tahun 2003 oleh sekelompok insinyur, termasuk Martin Eberhard dan Marc Tarpenning, Tesla bertujuan untuk membuktikan bahwa kendaraan listrik tidak hanya ramah lingkungan tetapi juga menawarkan performa yang lebih baik dibandingkan mobil berbahan bakar fosil. Nama "Tesla" sendiri diambil dari Nikola Tesla, seorang ilmuwan terkenal yang berkontribusi besar dalam pengembangan teknologi listrik.

Sejak penawaran saham perdana (IPO) pada tahun 2010, harga saham Tesla (TSLA) telah mengalami fluktuasi yang signifikan. Pada awalnya, saham ini bergerak relatif datar, namun mulai menunjukkan lonjakan dramatis sejak pertengahan 2013. Lonjakan paling mencolok terjadi pada tahun 2020, ketika harga saham meningkat lebih dari 700% akibat laporan keuangan yang kuat dan ekspektasi pasar yang tinggi terhadap masa depan kendaraan listrik. Pada saat itu, Tesla berhasil mencatatkan laba bersih dalam empat kuartal berturut-turut untuk pertama kalinya dalam sejarahnya.

Di bawah kepemimpinan Elon Musk sebagai CEO, Tesla telah berkembang pesat dengan memperkenalkan berbagai model kendaraan seperti Model S, Model 3, Model X, dan Model Y. Perusahaan ini juga berinvestasi dalam teknologi energi terbarukan, termasuk produk seperti solar roof dan Powerwall. Dengan visi untuk mempercepat transisi dunia ke energi berkelanjutan, Tesla tidak hanya fokus pada mobil listrik tetapi juga pada solusi energi yang lebih luas.

Meskipun mengalami beberapa tantangan dan penurunan harga saham di tahun-tahun tertentu—seperti penurunan sebesar 36% pada tahun 2022—Tesla tetap menjadi salah satu produsen mobil paling berharga di dunia. Perusahaan ini memiliki kapitalisasi pasar yang jauh lebih besar dibandingkan dengan pesaingnya di industri otomotif, seperti Toyota. Dengan terus berinovasi dan memperluas jangkauan produknya secara global, Tesla tetap menjadi pusat perhatian bagi investor dan konsumen di seluruh dunia.

1. Preprocessing Data
   a. Memeriksa Tipe Data

Setelah melakukan pembacaan dataset melalui file csv, kami memeriksa tipe data dengan show().

```
import pyspark.sql.functions as F
from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("CekTipeData").getOrCreate()

df = spark.read.csv("/content/TESLA.csv", header=True, inferSchema=True)

df.printSchema()
```

```
root
 |-- _c0: integer (nullable = true)
 |-- Date: string (nullable = true)
 |-- Open: double (nullable = true)
 |-- High: double (nullable = true)
 |-- Low: double (nullable = true)
 |-- Close: double (nullable = true)
 |-- Adj Close: double (nullable = true)
 |-- Volume: integer (nullable = true)
```

   b. Mengganti Nama Kolom

Kami mencoba mengganti nama kolom yang sudah ada

```
df = df.withColumnRenamed("c0", "col0") \
       .withColumnRenamed("Date", "tgl") \
       .withColumnRenamed("Open", "buka") \
       .withColumnRenamed("High", "tinggi") \
       .withColumnRenamed("Low", "rendah") \
       .withColumnRenamed("Close", "tutup") \
       .withColumnRenamed("Adj Close", "adj_tutup") \
       .withColumnRenamed("Volume", "vol")
df.printSchema()
```

```
root
 |-- _c0: integer (nullable = true)
 |-- tgl: string (nullable = true)
 |-- buka: double (nullable = true)
 |-- tinggi: double (nullable = true)
 |-- rendah: double (nullable = true)
 |-- tutup: double (nullable = true)
 |-- adj_tutup: double (nullable = true)
 |-- vol: integer (nullable = true)
```

   c. Melihat data

Melihat data pada data csv dengan df.show()

```
+---+--------+-----------+-----------+-----------+-----------+-----------+---------
|_c0|     tgl|       buka|     tinggi|     rendah|      tutup|  adj_tutup|      vol
+---+--------+-----------+-----------+-----------+-----------+-----------+---------
|  0| 6/29/10|1.266667008|1.666666985|1.169332981|1.592666984|1.592666984|281494500
|  1| 6/30/10|1.719333053|2.028000116|1.553333044|1.588667035|1.588667035|257806500
|  2|  7/1/10|1.666666985|1.728000045|1.351333022|1.463999987|1.463999987|123282000
|  3|  7/2/10|1.533332944|1.539999962| 1.24733305|1.279999971|1.279999971| 77097000
|  4|  7/6/10|1.333333015|1.333333015|1.055333018|1.074000001|1.074000001|103003500
|  5|  7/7/10|1.093333006|1.108667016|0.998667002|1.053333044|1.053333044|103825500
|  6|  7/8/10|1.075999975|1.167999983|1.037999988|1.164000034|1.164000034|115671000
|  7|  7/9/10|1.172000051| 1.19333303|1.103332996|1.159999967|1.159999967| 60759000
|  8| 7/12/10|1.196666956|1.204666972|1.133332968|1.136667013|1.136667013| 33037500
|  9| 7/13/10|1.159332991| 1.24266696|1.126667023|1.209332943|1.209332943| 40201500
| 10| 7/14/10| 1.19599998|1.343333006|1.184000015|1.322667003|1.322667003| 62928000
| 11| 7/15/10|1.329332948|1.433333039|1.266667008|1.325999975|1.325999975| 56097000
| 12| 7/16/10|1.379999995|1.419999957|1.336666942|1.376000047|1.376000047| 39319500
| 13| 7/19/10|1.424667001|1.483332992|1.394667029|1.460667014|1.460667014| 37297500
| 14| 7/20/10|1.456666946|1.456666946|1.336666942|1.353332996|1.353332996| 27379500
```

d. Memeriksa Nilai Null

Memeriksa nilai Null pada data

```
import pyspark.sql.functions as F

# Menghitung jumlah nilai null pada setiap kolom
df.select([F.count(F.when(F.isnull(c), c)).alias(c) for c in df.columns]).show()

# Menampilkan baris yang mengandung nilai null pada kolom "tgl"
df.filter(F.isnull(F.col("tgl"))).show()
```

```
+---+---+----+------+------+-----+---------+---+
|_c0|tgl|buka|tinggi|rendah|tutup|adj_tutup|vol|
+---+---+----+------+------+-----+---------+---+
|  0|  0|   0|     0|     0|    0|        0|  0|
+---+---+----+------+------+-----+---------+---+

+---+---+----+------+------+-----+---------+---+
|_c0|tgl|buka|tinggi|rendah|tutup|adj_tutup|vol|
+---+---+----+------+------+-----+---------+---+
+---+---+----+------+------+-----+---------+---+
```

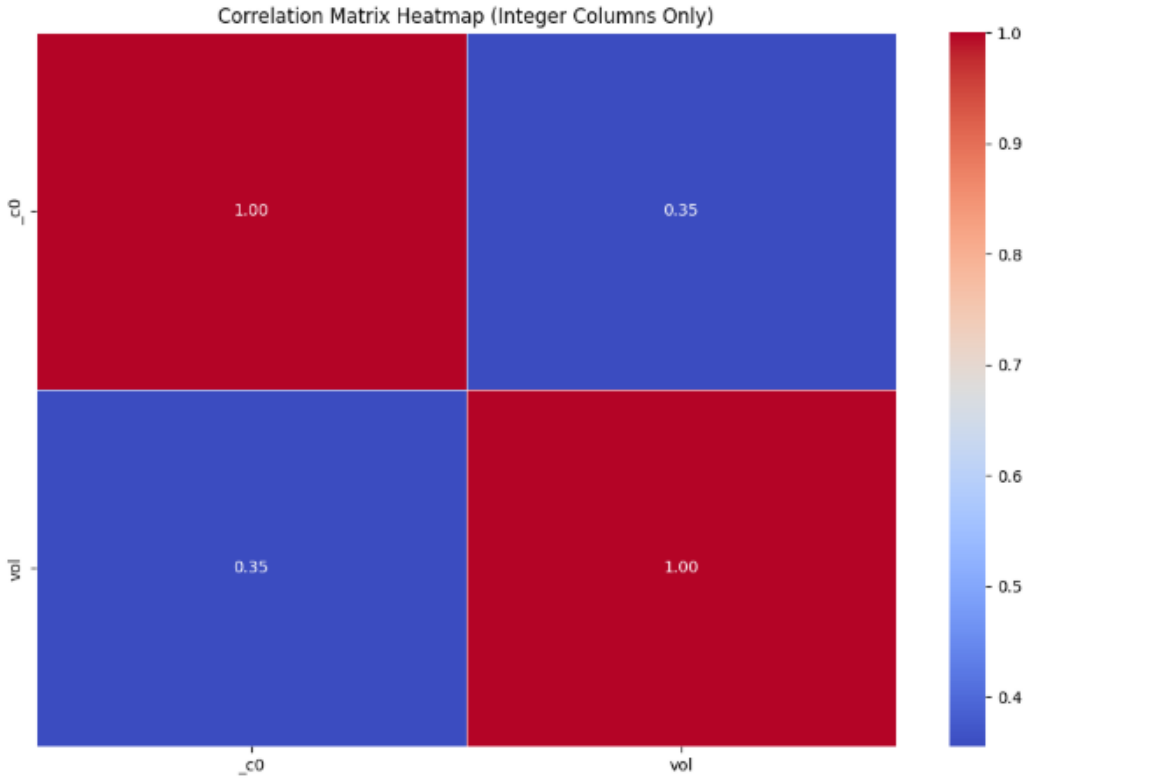
e. Menampilkan Nilai Sumary

```
import pyspark.sql.functions as F

df.describe().show()
```

summary	_c0	tgl	buka	tinggi	rendah	tutup	adj_tutup	vol
count	3637	3637	3637	3637	3637	3637	3637	3637
mean	1818.0	NULL	80.08057396726478	81.8327098768875	78.21927645228948	80.06795196634039	80.06795196634039	9.667329755292824E7
stddev	1050.055792168524	NULL	105.46613092977255	107.8071696743339	102.93120789566268	105.40944792971469	105.40944792971469	7.787314294808559E7
min	0	1/10/11	1.075999975	1.108667016	0.998667002	1.053333044	1.053333044	1777500
max	3636	9/9/24	411.4700012	414.4966736	405.6666565	409.9700012	409.9700012	914082000

f. Menampilkan Matriks Korelasinya

```
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.stat import Correlation
import seaborn as sns
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.stat import Correlation
import seaborn as sns
import matplotlib.pyplot as plt

integer_columns = [col_name for col_name, col_type in df.dtypes if col_type == 'int'
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.stat import Correlation
import seaborn as sns
import matplotlib.pyplot as plt

integer_columns = [col_name for col_name, col_type in df.dtypes if col_type == 'int'
vector_assembler = VectorAssembler(inputCols=integer_columns, outputCol="features")
df_assembled = vector_assembler.transform(df).select("features")
matrix = Correlation.corr(df_assembled, "features").head()
correlation_matrix = matrix[0].toArray()
column_names = df_assembled.schema["features"].metadata["ml_attr"]["attrs"]["numeric'
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", fmt=".2f", linewidths=.5
            xticklabels=[col["name"] for col in column_names],
            yticklabels=[col["name"] for col in column_names])
plt.title("Correlation Matrix Heatmap (Integer Columns Only)")
plt.show()
```

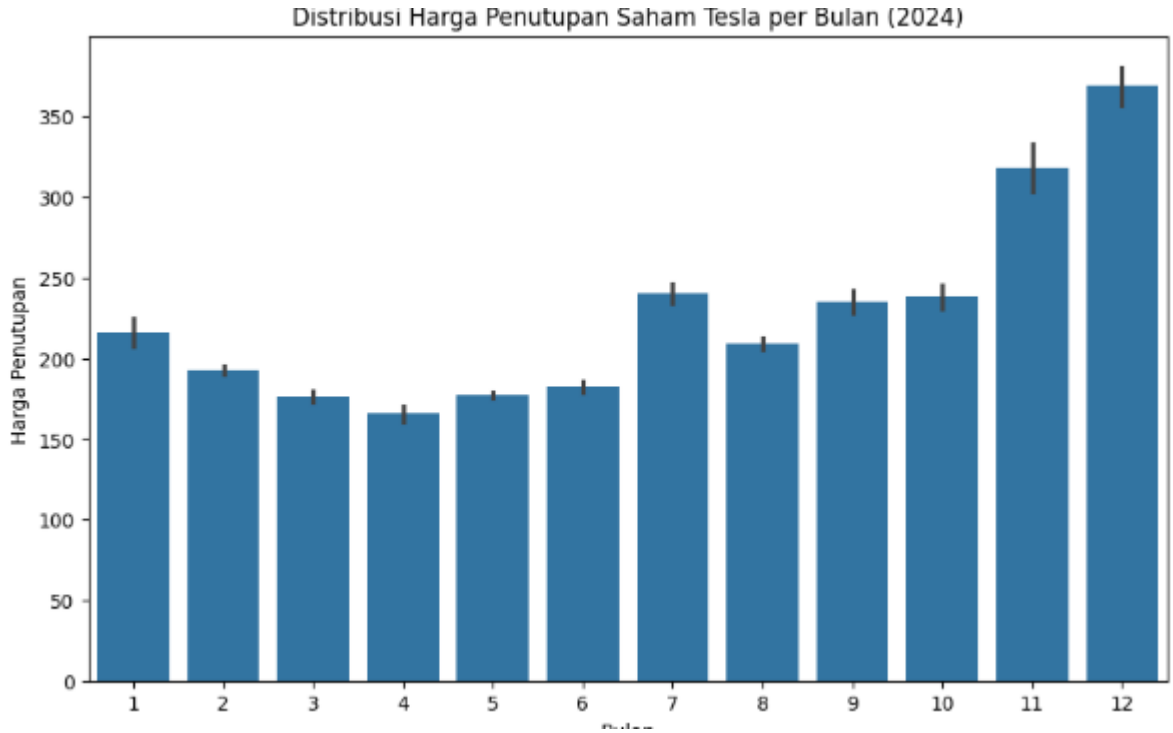


g. exploratory data analysis (EDA) bar dan pie

BAR.

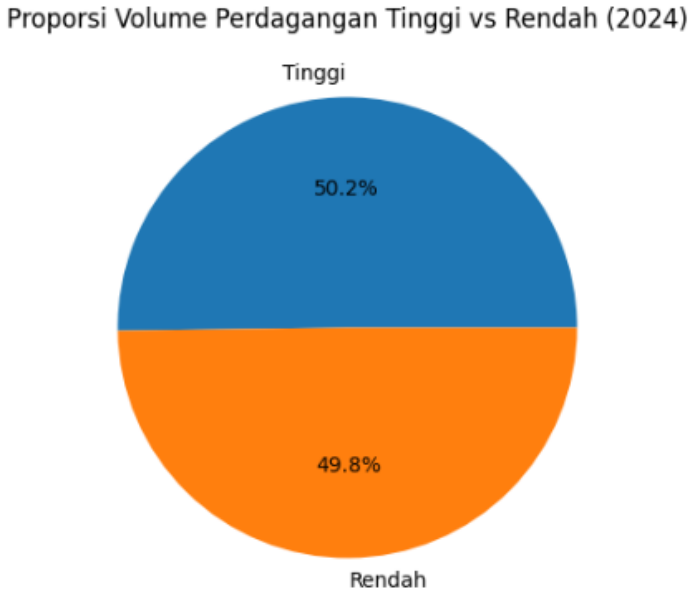
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df = pd.read_csv('/content/TESLA.csv', parse_dates=['Date'])

# Filter data untuk tahun 2024
df_2024 = df[df['Date'].dt.year == 2024]

# Analisis deskriptif
print(df_2024.describe())

# Visualisasi menggunakan bar chart
# Distribusi harga penutupan per bulan
plt.figure(figsize=(10, 6))
sns.barplot(x=df_2024['Date'].dt.month, y='Close', data=df_2024)
plt.title('Distribusi Harga Penutupan Saham Tesla per Bulan (2024)')
plt.xlabel('Bulan')
plt.ylabel('Harga Penutupan')
plt.show()
```

```
        Unnamed: 0                           Date        Open        High  \
count   237.000000                            237  237.000000  237.000000
mean   3518.000000  2024-06-20 23:53:55.443037952  217.609579  222.198439
min    3400.000000            2024-01-02 00:00:00  140.559998  144.440002
25%    3459.000000            2024-03-27 00:00:00  179.990005  183.259995
50%    3518.000000            2024-06-21 00:00:00  208.630005  213.190002
75%    3577.000000            2024-09-16 00:00:00  241.809998  246.210007
max    3636.000000            2024-12-09 00:00:00  397.609985  404.799988
std      68.560193                            NaN   50.015113   51.423516

              Low       Close   Adj Close        Volume
count  237.000000  237.000000  237.000000  2.370000e+02
mean   213.061603  217.699241  217.699241  9.468827e+07
min    138.800003  142.050003  142.050003  3.716760e+07
25%    175.580002  178.789993  178.789993  7.098810e+07
50%    204.820007  207.830002  207.830002  8.675950e+07
75%    236.320007  240.660004  240.660004  1.098157e+08
max    378.010010  389.790008  389.790008  2.438697e+08
std     48.490013   50.158618   50.158618  3.373331e+07
```



PIE.

```
volume_median = df_2024['Volume'].median()
df_2024.loc[df_2024['Volume'] > volume_median, 'VolumeCategory'] = 'Tinggi'
df_2024.loc[df_2024['Volume'] <= volume_median, 'VolumeCategory'] = 'Rendah'


plt.figure(figsize=(5, 5))
labels = df_2024['VolumeCategory'].unique()
plt.pie(df_2024['VolumeCategory'].value_counts(), labels=labels, autopct='%1.1f%%')
plt.title('Proporsi Volume Perdagangan Tinggi vs Rendah (2024)')
plt.show()
```



2. Pengembangan Model Machine Learning

a. Model Machine Learning

Penggunaan 4 model machine learning pada library PySpark yaitu Random Forest, Gradient Boost Tree, Logistic Regression, dan Decision Tree Classifier

- Random Forest

```
Accuracy: 0.5718050065876152
```

- Gradient Boost Tree

```
GBT Accuracy: 0.61133069828722
```

- Logistic Regression

```
Logistic Regression Accuracy: 0.9960474308300395
```

- Decision Tree Classifier

```
Decision Tree Accuracy: 0.5177865612648221
```

Hasil dari membandingkan menggunakan metriks seperti AUC (ROC Curve), Akurasi, F1 Score, Presisi, dan Recall.

```
print(f"AUC: {auc}")
print(f"Akurasi: {accuracy}")
print(f"Presisi: {precision}")
print(f"Recall: {recall}")
print(f"F1 Score: {f1_score}")
```

```
AUC: 0.6146958755728377
Akurasi: 0.5718050065876152
Presisi: 0.5718398530766604
Recall: 0.5718050065876152
F1 Score: 0.5717737864750836
```

b. Mengclassification 2 model dengan performa terbaik

- Random Forest

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

# Definisikan Logistic Regression
lr = LogisticRegression(labelCol="label", featuresCol="features")

# Buat grid hyperparameter untuk diuji
param_grid = (ParamGridBuilder()
              .addGrid(lr.regParam, [0.01, 0.1, 1.0])  # Regularization parameter
              .addGrid(lr.elasticNetParam, [0.0, 0.5, 1.0])  # Elastic Net mixing par
              .build())

# Evaluator untuk menghitung performa model
evaluator = BinaryClassificationEvaluator(labelCol="label", metricName="areaUnderROC'

# Definisikan CrossValidator
cross_validator = CrossValidator(estimator=lr,
                                 estimatorParamMaps=param_grid,
                                 evaluator=evaluator,
                                 numFolds=5)  # Cross-validation dengan 5 fold
```

Hasil

```
Logistic Regression Accuracy (Best Model): 0.6132886187382101
Best Model Parameters:
  regParam: 0.01
  elasticNetParam: 0.0
```

- Logistic Regression

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

# Definisikan GBTClassifier
gbt = GBTClassifier(labelCol="label", featuresCol="features", seed=42)

# Buat grid hyperparameter untuk diuji
param_grid = (ParamGridBuilder()
              .addGrid(gbt.maxDepth, [3, 5, 7])  # Kedalaman maksimum pohon
              .addGrid(gbt.maxIter, [10, 20, 30])  # Jumlah iterasi boosting
              .addGrid(gbt.stepSize, [0.1, 0.2, 0.3])  # Learning rate
              .build())

# Evaluator untuk menghitung performa model
evaluator = BinaryClassificationEvaluator(labelCol="label", metricName="areaUnderROC'

# Definisikan CrossValidator
cross_validator = CrossValidator(estimator=gbt,
                                 estimatorParamMaps=param_grid,
                                 evaluator=evaluator,
                                 numFolds=5)  # Cross-validation dengan 5 fold
```

Hasil

```
GBT Accuracy (Best Model): 0.696818277090757
Best Model Parameters:
  maxDepth: 5
  maxIter: 30
  stepSize: 0.3
```

c. Model terbaik yang saya dapatkan

```
# Gunakan model terbaik untuk prediksi pada data uji
best_model = cv_model.bestModel
gbt_predictions = best_model.transform(test_data)

# Evaluasi model terbaik
gbt_accuracy = evaluator.evaluate(gbt_predictions)
print(f"GBT Accuracy (Best Model): {gbt_accuracy}")

# Hyperparameter terbaik
print("Best Model Parameters:")
print(f"  maxDepth: {best_model.getOrDefault('maxDepth')}")
print(f"  maxIter: {best_model.getOrDefault('maxIter')}")
print(f"  stepSize: {best_model.getOrDefault('stepSize')}")
```

```
GBT Accuracy (Best Model): 0.696818277090757
Best Model Parameters:
  maxDepth: 5
  maxIter: 30
  stepSize: 0.3
```

Pada model di atas menunjukkan proses evaluasi model Gradient Boosting Tree (GBT) setelah dilakukan penyetelan hyperparameter. Model GBT terbaik yang diperoleh dari proses tuning kemudian digunakan untuk memprediksi data uji. Akurasi prediksi model terbaik ini kemudian dihitung dan ditampilkan. Selain itu, kode juga menampilkan nilai hyperparameter terbaik yang digunakan oleh model, yaitu maxDepth, maxIter, dan stepSize. Nilai-nilai ini menunjukkan kedalaman maksimum pohon, jumlah iterasi, dan ukuran langkah pembelajaran yang optimal untuk model GBT tersebut. Hasil evaluasi menunjukkan bahwa model GBT dengan konfigurasi hyperparameter yang telah ditentukan dapat memprediksi dengan akurasi sekitar 69,68% pada data uji.

d. Lampiran
   - Link Github
     https://github.com/Habibcool/Harga-saham-pada-tesla
   - Link Launchinpad
     https://launchinpad.com/project/harga-saham-pada-tesla-86f6f22
   - Job Description Anggota Kelompok

KONTRIBUSI (JOBDESK) ANGGOTA KELOMPOK

Nama	Kontribusi
22.11.4571_Habib Baitul Hamdi	Preprocessing Data & EDA & Eksplorasi Dataset
22.11.4598_Zahran Nugraha	Pengembangan Model & Hyperparameter Tuning