Memprediksi Peningkatan Pemasaran Bank

LAPORAN TECHNICAL REPORT

DISUSUN OLEH :

KELOMPOK 5

ANALICIA	**22031554007**
AHMAD RAFI SYAIFUDIN	**22031554030**
RIVA DIAN ARDIANSYAH	**22031554043**

S1 SAINS DATA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS NEGERI SURABAYA

2023

DAFTAR ISI

BAB I
PENDAHULUAN

1.1 LATAR BELAKANG

Project ini membahas berbagai jenis model algoritma untuk pengklasifikasian dengan menggunakan beberapa model, yaitu Random Forest, Decission Tree, dan Logistic Regression. Dengan tujuan untuk mengevaluasi dan menerapkan metode tersebut. Data yang digunakan berisi pemasaran suatu institusi bank. Project ini memprediksi terhadap suatu klien bank yang akan berlangganan deposito berjangka panjang (variabel y) berdasarkan atribut pemasaran oleh institusi bank.

1.2 Tujuan dan Manfaat

Ada pun project ini dilakukan ialah bertujuan untuk melakukan klasifikasi terhadap klien bank yang terlihat memungkinkan untuk berlanganan deposito yang berjangka Panjang. Data bank yang digunakan masih tergolong inbalance sehingga dilakukannya resampling pada data. Kemudian data tersebut dilakukan tuning agar mendapat performa akurasi yang lebih baik dari sebelumnya.

Dalam hal ini dapat memberikan manfaat untuk meningkatkan efisiensi pemasaran sehingga bank dapat memberikan arahan pemasaran yang tepat. Manfaat dilakukannnya pengklasifikasian menggunakan ketiga algoritma tersebut ialah melihat prediksi yang dilakukan akurat untuk membandingkan dan mengevaluasi kinerja algoritma. Selain itu, juga dapat melihat bagaimana penggunaan informasi yang dikembangkan berdasarkan strategi pemasaran terhadap pelanggan maupun kebutuhan pelanggan tersebut.

BAB II
METODE

2.1 RANDOM FOREST

Random Forest merupakan teknik dalam machine learning yang menggunakan metode pohon keputusan untuk mengevaluasi model sehingga melalui pohon keputusan ini digunakan untuk membuat keputusan yang memperlihatkan kondisi dari tiap cabang tersebut. Random forest sendiri memiliki set data latih yang berbeda untuk mengurangi overfitting dan meningkatkan ketahanan pada model itu sendiri.

Pada random forest dapat digunakan untuk menganalisis faktor dan strategi pemasaran bank terhadap pelanggan. Selain itu, dapat membantu dalam memprediksi risiko pelanggan terhadap deposito sehingga dari pihak bank dapat membuat keputusan yang lebih baik. Maka random forest ini dapat digunakan untuk memprediksi dan analisis terhadap peningkatan strategi pemasaran.

2.2 DECISSION TREE

Decision tree merupakan machine learning yang juga digunakan untuk memprediksi atau mengklasifikasi data dengan mengambil keputusan dengan cara yang mirip dengan pohon kepututsan atau random forest. Decision tree mudah melakukan interpretasi dan visualisasi yang memudahkan dalam pengambilan keputusan, juga tidak memiliki tahap preprocessing yang rumit dan memiliki fitur otomatis dengan cara mengidentifikasi fitur yang penting dalam pembuatan keputusan.

Decision tree juga memiliki batasan yang dapat menyebabkan overfitting dan menjadi tidak stabil karena memiliki perubahan yang kecil didalam data. Meskipun demikian, machine learning ini tetap baik digunakan karena mudah dalam interpretasi dan pengklasifikasiannya yang relative sederhana.

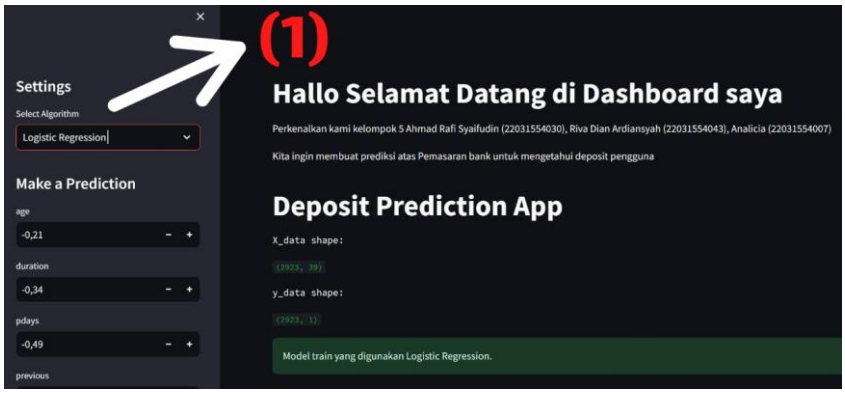
2.3 LOGISTIC REGRESSION

Logistic regression adalah suatu model statistik umum tentunya digunakan untuk menganalisis. Logistic regression digunakan untuk memodelkan suatu hubungan antara variable dependen yang memiliki dua nilai yang mungkin, seperti ya atau tidak, 0 atau 1, dan lain-lain. Logistic regression juga digunakan untuk memahami faktor-faktor yang signifikan terhadap strategi dalam pemasaran.

Logistic regression ini sendiri memiliki interpretasi yang mudah dalam bebagai kasus, salah satunya pemasaran bank. Logistic regression juga merupakan suastu metode yang baik untuk memprediksi probabilitas, memiliki performa yang baik untuk data yang telah terstruktur. Dalam pemasaran bank ini salah satu metode yang sesuai sehingga dapa tmenjadi alat yang efisien untuk analisis prediktif.

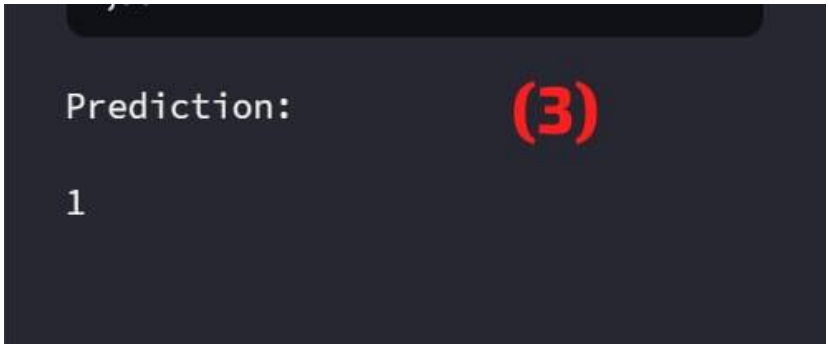
BAB III
HASIL DAN ANALISIS

1.1 USER MANUAL GUIDE



(1) Dapat dilihat pada tahap pertama dengan memilih algoritma yang ingin digunakan, yaitu Random Forest, Decision Tree, dan Logistic Regression

(2) Pada tahap yang kedua ini dapat mengukur prediction yang ingin digunakan



(3) Tahap ketiga ini menampilkan hasil dari model dan pengukuran yang telah kita gunakan sehingga akan mengeluarkan hasil 1 untuk “yes” dan 0 untuk “no”.

1.2 LISTING CODE

```
[4]
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import f1_score
from sklearn.metrics import average_precision_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix, classification_report, precision_score, recall_score
```

(1) Diatas adalah library yang digunakan dalam project ini.

```
[10]
# Memeriksa dimensi dari dataset (jumlah baris, jumlah kolom)
df.shape
```

```
(45211, 17)
```

(2) Pada gambar diatas menggunakan dataset ‘’bank-full.csv’’ diketahui 45.211 data dengan 17 kolom.

```
[12]
# Memeriksa apakah ada duplikasi dalam data
df.duplicated().sum()
```

```
0
```

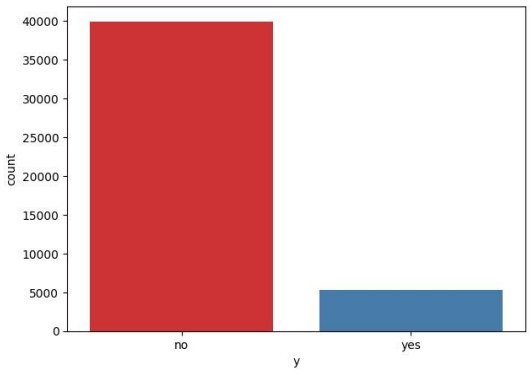
```
[13]
# Memeriksa apakah terdapat missing values dalam data
df.isnull().sum()
```

```
age          0
job          0
marital      0
education    0
default      0
balance      0
housing      0
loan         0
contact      0
day          0
month        0
duration     0
campaign     0
pdays        0
previous     0
poutcome     0
y            0
dtype: int64
```

(3) Diatas dilakukan data cleaning dengan memeriksa apakah terdaapt duplikasi data dan missing value.

```
[17]  df['y'].value_counts()
```

```
y
no     39922
yes     5289
Name: count, dtype: int64
```



(4) Dengan memeriksa frekuensi nilai pada variabel y diketahui ‘’no’’ sebanyak 39.922 dan ‘’yes’’ sebanyak 5.289 dan disebelah kanan terdapat visualisasi data yang terlihat ketimpangan antara nilai ‘’no’’ dan ‘’yes’’ tersebut.

```
import pandas as pd

# Load the dataset
df = pd.read_csv("bank-full.csv", sep=";")

# Check unique values in the 'y' column
print(df['y'].value_counts())

# Resample the data for 'no' values in the 'y' column to have 5000 occurrences
df_resampled = pd.concat([df[df['y'] == 'yes'], df[df['y'] == 'no'].sample(5300, replace=True)])

# Check the unique values again to confirm the resampling
print(df_resampled['y'].value_counts())

# Save the resampled DataFrame to a new CSV file
df_resampled.to_csv("bankresampling_coy.csv", index=False)
```

[4] ✓ 0.1s

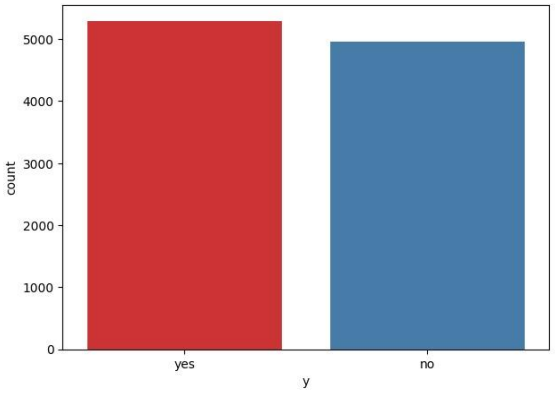
```
y
no     39922
yes     5289
Name: count, dtype: int64
y
no     5300
yes    5289
Name: count, dtype: int64
```

(5) Pada code diatas terdapat code resample untuk membuat data tersebut dari inbalance menjadi balance, dimana ‘’no’’ sebanyak 5.300 dan ‘’yes’’ sebanyak 5.289.

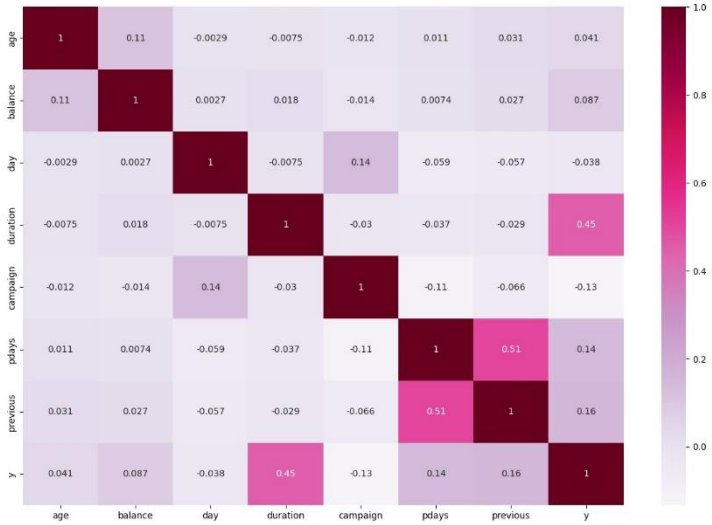
```
[17]  df['y'].value_counts()
```

✓ 0.0s

```
y
yes    5289
no     4952
Name: count, dtype: int64
```



(6) Ini merupakan nilai frekuensi y setelah di resample.



(7) Dapat dilihat korelasinya terhadap variabel y menggunakan heatmap.

```
df.drop(['month'], axis=1, inplace=True)
```

(8) Kami melakukan preprocessing dnegan penghapusan kolom yang tidak berpengaruh terhadap variabel y.

```
df_scalling['job'] = df['job']
df_scalling['marital'] = df['marital']
df_scalling['education'] = df['education']
df_scalling['default'] = df['default']
df_scalling['housing'] = df['housing']
df_scalling['loan'] = df['loan']
df_scalling['contact'] = df['contact']
df_scalling['poutcome'] = df['poutcome']
```

```
# Encoder
df_new = pd.get_dummies(df_scalling)
```

```
df_new.head()
```

	age	duration	pdays	previous	balance	campaign	day	job_admin.	job_blue-collar	job_entrepreneur	...	loan_no	loan
0	1.758260	1.841148	-0.492707	-0.384698	0.280071	-0.752669	-1.224573	1	0	0	...	1	
1	1.476893	3.034172	-0.492707	-0.384698	-0.475177	-0.752669	-1.224573	1	0	0	...	1	
2	0.070053	2.815217	-0.492707	-0.384698	-0.072575	-0.752669	-1.224573	0	0	0	...	1	
3	1.383103	0.541454	-0.492707	-0.384698	0.323782	-0.752669	-1.224573	0	0	0	...	1	
4	1.289314	0.805323	-0.492707	-0.384698	-0.429494	-0.123075	-1.224573	1	0	0	...	1	

5 rows × 40 columns

(9) Kemudian data tersebut kami encoder untuk disimpan dalam format tertentu agar mudah dianalisis.

```
X = df_new.drop('y', axis=1)
y = df_new.y
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

(10) Data tersebut di split.

```
lr = LogisticRegression()
tree = DecisionTreeClassifier
rf = RandomForestClassifier()
```

```
# Logistic Regression
lr.fit(X_train, y_train)
y_pred_lr = lr.predict(X_test)
```

```
# Decision Tree
tree.fit(X_train, y_train)
y_pred_tree = tree.predict(X_test)
```

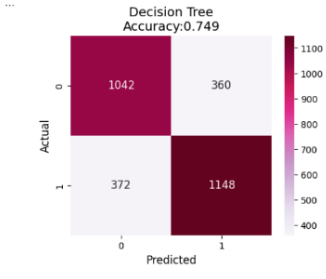
```
# Random Forest
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)
```

(11) Pada code diatas dilakukannya pendefinisian dan pelatihan model.

```
# Create a confusion Matrix for Logistic Regression
cm = confusion_matrix(y_test, y_pred_lr)

plt.figure(figsize=(5,4))
sns.heatmap(cm, annot=True,fmt=".0f", cmap="PuRd", annot_kws={"fontsize":12})
plt.title("Logistic Regression\nAccuracy:{:.3f}".format(accuracy_score(y_test, y_pred_lr)), fontsize=14)
plt.xlabel("Predicted", fontsize=12)
plt.ylabel("Actual", fontsize=12)
plt.show()
print("Classification Report:\n", classification_report(y_test, y_pred_lr))

# print(f'Accuracy Score:- {accuracy_score(y_test, y_pred_lr)}')
# print(f'Precision Score:- {precision_score(y_test,y_pred_lr)}')
# print(f'Recall Score:- {recall_score(y_test,y_pred_lr)}')
```

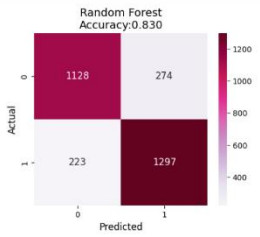

```
Classification Report:
              precision    recall  f1-score   support

           0       0.79      0.84      0.82      1402
           1       0.85      0.80      0.82      1520

    accuracy                           0.82      2922
   macro avg       0.82      0.82      0.82      2922
weighted avg       0.82      0.82      0.82      2922
```

```
# Create a confusion Matrix for Decision Tree
cm = confusion_matrix(y_test, y_pred_tree)

plt.figure(figsize=(5,4))
sns.heatmap(cm, annot=True,fmt=".0f", cmap="PuRd", annot_kws={"fontsize":12})
plt.title("Decision Tree\nAccuracy:{:.3f}".format(accuracy_score(y_test, y_pred_tree)), fontsize=14)
plt.xlabel("Predicted", fontsize=12)
plt.ylabel("Actual", fontsize=12)
plt.show()
print("Classification Report:\n", classification_report(y_test, y_pred_tree))

# print(f'Accuracy Score:- {accuracy_score(y_test, y_pred_tree)}')
# print(f'Precision Score:- {precision_score(y_test,y_pred_tree)}')
# print(f'Recall Score:- {recall_score(y_test,y_pred_tree)}')
```



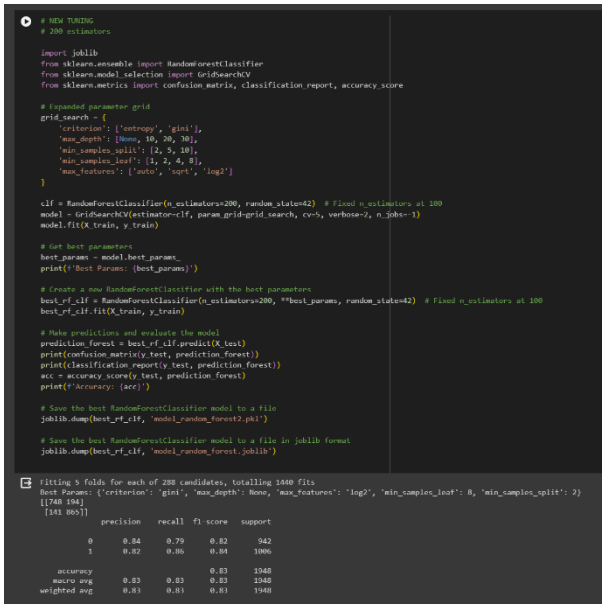

```
Classification Report:
              precision    recall  f1-score   support

           0       0.74      0.74      0.74      1402
           1       0.76      0.76      0.76      1520

    accuracy                           0.75      2922
   macro avg       0.75      0.75      0.75      2922
weighted avg       0.75      0.75      0.75      2922
```

```
#  Create a confusion Matrix for Random Forest
cm = confusion_matrix(y_test, y_pred_rf)

plt.figure(figsize=(5,4))
sns.heatmap(cm, annot=True,fmt=".0f", cmap="PuRd", annot_kws={"fontsize":12})
plt.title("Random Forest\nAccuracy:{:.3f}".format(accuracy_score(y_test, y_pred_rf)), fontsize=14)
plt.xlabel("Predicted", fontsize=12)
plt.ylabel("Actual", fontsize=12)
plt.show()
print("Classification Report:\n", classification_report(y_test, y_pred_rf))

print(f'Accuracy Score:- {accuracy_score(y_test, y_pred_rf)}')
print(f'Precision Score:- {precision_score(y_test,y_pred_rf)}')
print(f'Recall Score:- {recall_score(y_test,y_pred_rf)}')
```




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

           0       0.83      0.80      0.82      1402
           1       0.83      0.85      0.84      1520

    accuracy                           0.83      2922
   macro avg       0.83      0.83      0.83      2922
weighted avg       0.83      0.83      0.83      2922
```

(12) Diatas merupakan hasil evaluasi dari ketiga model

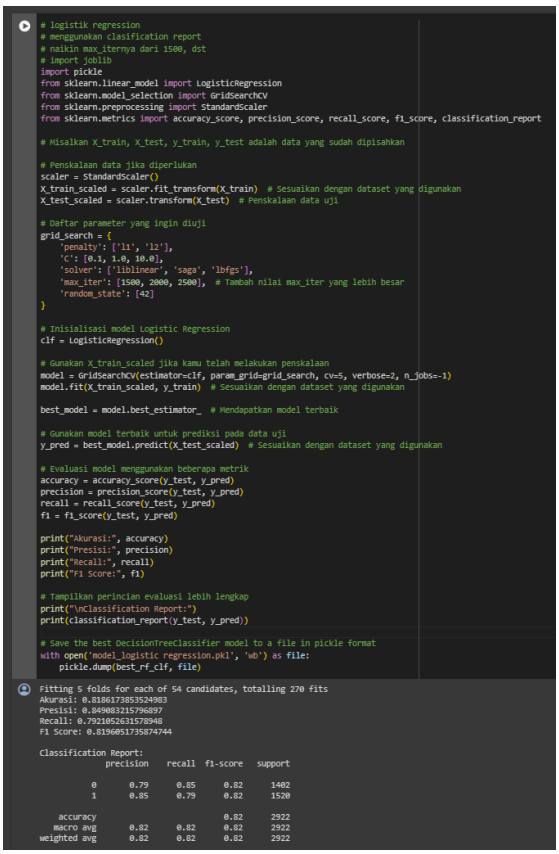


```
# NEW TUNING
# 200 estimators

import joblib
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score

# Expanded parameter grid
grid_search = {
    'criterion': ['entropy', 'gini'],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4, 8],
    'max_features': ['auto', 'sqrt', 'log2']
}

clf = RandomForestClassifier(n_estimators=200, random_state=42)  # Fixed n_estimators at 100
model = GridSearchCV(estimator=clf, param_grid=grid_search, cv=5, verbose=2, n_jobs=-1)
model.fit(X_train, y_train)

# Get best parameters
best_params = model.best_params_
print(f'Best Params: {best_params}')

# Create a new RandomForestClassifier with the best parameters
best_rf_clf = RandomForestClassifier(n_estimators=200, **best_params, random_state=42)  # Fixed n_estimators at 100
best_rf_clf.fit(X_train, y_train)

# Make predictions and evaluate the model
prediction_forest = best_rf_clf.predict(X_test)
print(confusion_matrix(y_test, prediction_forest))
print(classification_report(y_test, prediction_forest))
acc = accuracy_score(y_test, prediction_forest)
print(f'Accuracy: {acc}')

# Save the best RandomForestClassifier model to a file
joblib.dump(best_rf_clf, 'model_random_forest2.pkl')

# Save the best RandomForestClassifier model to a file in joblib format
joblib.dump(best_rf_clf, 'model_random_forest.joblib')
```

```
Fitting 5 folds for each of 288 candidates, totalling 1440 fits
Best Params: {'criterion': 'gini', 'max_depth': None, 'max_features': 'log2', 'min_samples_leaf': 8, 'min_samples_split': 2}
[[748 194]
 [141 865]]
              precision    recall  f1-score   support

           0       0.84      0.79      0.82       942
           1       0.82      0.86      0.84      1006

    accuracy                           0.83      1948
   macro avg       0.83      0.83      0.83      1948
weighted avg       0.83      0.83      0.83      1948
```

```
# Decision tree
# menggunakan parameter sendiri & fiks
# import joblib
import pickle
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score

# Expanded parameter grid
grid_search = {
    # 'n_estimators' : [100,200,300],
    'criterion': ['entropy', 'gini'],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4, 8],
    'max_features': ['auto', 'sqrt', 'log2']
}

clf = DecisionTreeClassifier(random_state=42)
model = GridSearchCV(estimator=clf, param_grid=grid_search, cv=5, verbose=2, n_jobs=-1)
model.fit(X_train, y_train)

# Get best parameters
best_params = model.best_params_
print(f'Best Params: {best_params}')

# Create a new RandomForestClassifier with the best parameters
best_rf_clf = DecisionTreeClassifier(**best_params, random_state=42)
best_rf_clf.fit(X_train, y_train)

# Make predictions and evaluate the model
prediction_forest = best_rf_clf.predict(X_test)
print(confusion_matrix(y_test, prediction_forest))
print(classification_report(y_test, prediction_forest))
acc = accuracy_score(y_test, prediction_forest)
print(f'Accuracy: {acc}')

# # Memuat kembali model dari file .pkl
# loaded_model = joblib.load('model_decision_tree.pkl')

# # Simpan kembali model yang dimuat dalam format .joblib
# joblib.dump(loaded_model, 'model_decision_tree.joblib')

# Save the best DecisionTreeClassifier model to a file in pickle format
with open('model_decision_tree.pkl', 'wb') as file:
    pickle.dump(best_rf_clf, file)
```

```
Fitting 5 folds for each of 288 candidates, totalling 1440 fits
Best Params: {'criterion': 'gini', 'max_depth': 10, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_samples_split': 2}
[[ 959  443]
 [ 226 1294]]
              precision    recall  f1-score   support

           0       0.80      0.68      0.74      1402
           1       0.74      0.84      0.79      1520

    accuracy                           0.77      2922
   macro avg       0.77      0.76      0.76      2922
weighted avg       0.77      0.77      0.77      2922
```

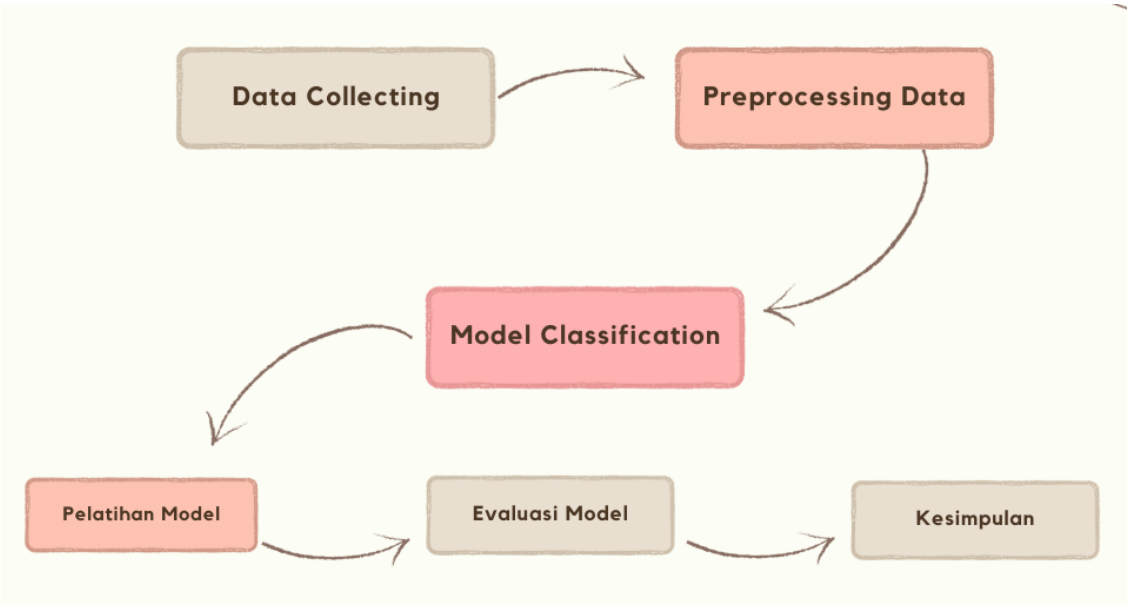


```
# logistik regression
# menggunakan clasification report
# naikin max_iternya dari 1500, dst
# import joblib
import pickle
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, classification_report

# Misalkan X_train, X_test, y_train, y_test adalah data yang sudah dipisahkan

# Penskalaan data jika diperlukan
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)  # Sesuaikan dengan dataset yang digunakan
X_test_scaled = scaler.transform(X_test)  # Penskalaan data uji

# Daftar parameter yang ingin diuji
grid_search = {
    'penalty': ['l1', 'l2'],
    'C': [0.1, 1.0, 10.0],
    'solver': ['liblinear', 'saga', 'lbfgs'],
    'max_iter': [1500, 2000, 2500],  # Tambah nilai max_iter yang lebih besar
    'random_state': [42]
}

# Inisialisasi model Logistic Regression
clf = LogisticRegression()

# Gunakan X_train_scaled jika kamu telah melakukan penskalaan
model = GridSearchCV(estimator=clf, param_grid=grid_search, cv=5, verbose=2, n_jobs=-1)
model.fit(X_train_scaled, y_train)  # Sesuaikan dengan dataset yang digunakan

best_model = model.best_estimator_  # Mendapatkan model terbaik

# Gunakan model terbaik untuk prediksi pada data uji
y_pred = best_model.predict(X_test_scaled)  # Sesuaikan dengan dataset yang digunakan

# Evaluasi model menggunakan beberapa metrik
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print("Akurasi:", accuracy)
print("Presisi:", precision)
print("Recall:", recall)
print("F1 Score:", f1)

# Tampilkan perincian evaluasi lebih lengkap
print("\nClassification Report:")
print(classification_report(y_test, y_pred))

# Save the best DecisionTreeClassifier model to a file in pickle format
with open('model_logistic regression.pkl', 'wb') as file:
    pickle.dump(best_rf_clf, file)
```

```
Fitting 5 folds for each of 54 candidates, totalling 270 fits
Akurasi: 0.8186173853524983
Presisi: 0.8490832157968897
Recall: 0.7921052631578948
F1 Score: 0.8196051735874744

Classification Report:
              precision    recall  f1-score   support

           0       0.79      0.85      0.82      1402
           1       0.85      0.79      0.82      1520

    accuracy                           0.82      2922
   macro avg       0.82      0.82      0.82      2922
weighted avg       0.82      0.82      0.82      2922
```

(13) Code diatas merupakan hasil tuning dari Random Forest, Naïve Bayes, dan KNN

1.3 DIAGRAM ALIR



DETAIL JOB PROJECT

1. Ahmad Rafi Syaifudin (22031554030)
   - Streamlit
   - Cleaning Data
   - PPT
   - Undersampling
   - Hyperparameter Tuning
   - EDA

2. Analicia (22031554007)
   - Modelling
   - Split Data
   - PPT
   - Laporan
   - Data Preprocessing
   - Hyperparameter Tuning

3. Riva Dian Ardiansyah (22031554043)
   - Streamlit
   - Data Collecting
   - PPT
   - Confusion Matrix
   - Feature Importance

BAB IV
KESIMPULAN

Dalam laporan project ini dengan melakukan prediksi peningkatan pemasaran bank. Dengan beberapa tahap dalam pengklasifikasi yang dilakukan, yaitu observing data, cleaning data, kemudian melakukan evaluasi model, dan hyperparameter tuning, melakukan evaluasi juga melihat confusion matrix nya hingga pada klasifikasi reportnya.

Berdasarkan hasil keseluruhan yang telah kami lakukan. Pada prediksi peningkatan pemasaran bank dengan menggunakan 3 algoritma diantaranya Random Forest, Logistic Regression, Decision Tree. Kami melihat perbedaan tiap model, sehingga menunjukan performa terbaik dari ketiga model algoritma. Dari ketiga algoritma didapatkan hasil terbaik dari Random Forest yang telah dilakukan resample dan tuning sehingga menghasilkan akurasi sebesar 83%.

REFERENSI

[1] O. Apampa, "Evaluation of Classification and Ensemble Algorithms for Bank Customer Marketing Response Prediction," *Journal of International Technology and Information Management*, vol. 25, no. 4, Jan. 2016, doi: 10.58729/1941-6679.1296.

[2] P. Appiahene, Y. M. Missah, and U. Najim, "Predicting Bank Operational Efficiency Using Machine Learning Algorithm: Comparative Study of Decision Tree, Random Forest, and Neural Networks," *Advances in Fuzzy Systems*, vol. 2020, 2020, doi: 10.1155/2020/8581202.

[3] F. M. Basysyar, A. R. Dikananda, and D. A. Kurnia, "Prediction of Bank Customer Potential Using Creative Marketing Based on Exploratory Data Analysis and Decision Tree Algorithm," *Ingenierie des Systemes d'Information*, vol. 27, no. 4, pp. 597–604, Aug. 2022, doi: 10.18280/isi.270409.

[4] F. Safarkhani and S. Moro, "Improving the accuracy of predicting bank depositor's behavior using a decision tree," *Applied Sciences (Switzerland)*, vol. 11, no. 19, Oct. 2021, doi: 10.3390/app11199016.

[5] M. C. Keshava, "Predictive Analysis on Bank Marketing Campaign using Machine Learning Algorithms," *Int J Res Appl Sci Eng Technol*, vol. 7, no. 4, pp. 2664–2668, Apr. 2019, doi: 10.22214/ijraset.2019.4483.

[6] J. Alexandra and K. P. Sinaga, "Machine Learning Approaches for Marketing Campaign in Portuguese Banks," *2021 3rd International Conference on Cybernetics and Intelligent System (ICORIS)*, Oct. 2021, doi: https://doi.org/10.1109/icoris52787.2021.9649623.

[7] F. M. Basysyar, A. R. Dikananda, and D. A. Kurnia, "Prediction of Bank Customer Potential Using Creative Marketing Based on Exploratory Data Analysis and Decision Tree Algorithm," *Ingénierie des systèmes d information*, vol. 27, no. 4, pp. 597–604, Aug. 2022, doi: https://doi.org/10.18280/isi.270409.

[8] S. Moro, P. Cortez, and P. Rita, "A data-driven approach to predict the success of bank telemarketing," *Decision Support Systems*, vol. 62, pp. 22–31, Jun. 2014, doi: https://doi.org/10.1016/j.dss.2014.03.001.

[9] L. Vongchalerm, "Analysis of predicting the success of the banking telemarketing campaigns by using machine learning techniques," *norma.ncirl.ie*, Jan. 31, 2022. https://norma.ncirl.ie/6337/ (accessed Dec. 22, 2023).