

Informasi Proyek

Judul Proyek : **Analisis dan Pemodelan Prediktif untuk Penentuan Keberhasilan Langganan Deposito Jangka Pendek (Studi Kasus: Bank Marketing Campaign)**

Nama : Rizqi Agus Sahputra

NIM: 234311053

Program Studi: Teknologi Rekayasa Perangkat Lunak

Mata Kuliah: Data Science

Dosen Pengampu: Gus Nanang Syaifuddin

Tahun Akademik: 2025/2026

Link GitHub Repository : <https://github.com/ininduu/DataScience-BankCampaign-.git>

Link Video Pembahasan: <https://youtu.be/h2rErXh-4r4?si=ACnx9gEqMCpsbQVQ>

1. Learning Outcome : Pada proyek ini, mahasiswa diharapkan dapat

- A. Memahami konteks masalah dan merumuskan problem statement secara jelas
- B. Melakukan analisis dan eksplorasi data (EDA) secara komprehensif
- C. Melakukan data preparation yang sesuai dengan karakteristik dataset
- D. Mengembangkan tiga model machine learning yang terdiri dari (WAJIB):
 - **Model baseline**
 - **Model machine learning / advanced**
 - **Model deep learning (WAJIB)**
- E. Menggunakan metrik evaluasi yang relevan dengan jenis tugas ML
- F. Melaporkan hasil eksperimen secara ilmiah dan sistematis
- G. Mengunggah seluruh kode proyek ke GitHub (WAJIB)
- H. Menerapkan prinsip software engineering dalam pengembangan proyek

2. PROJECT OVERVIEW

2.1. Latar Belakang

- Mengapa proyek ini penting?

Jawab : Proyek ini sangat penting karena berfokus pada efisiensi dan efektivitas kampanye tentang pemasaran secara langsung untuk produk perbankan, khususnya pada deposito berjangka. Di tengah persaingan industri bank yang ketat, bank perlu menyesuaikan dana yang dialokasikan dengan benar untuk strategi pemasaran mereka, khususnya pada deposito berjangka dimana itu diprioritaskan kepada calon nasabah yang memiliki probabilitas tinggi kepada untuk berlangganan. Model ini akan digunakan untuk memprioritaskan nasabah target nasabah.

- Permasalahan umum pada domain terkait (misal: kesehatan, pendidikan, keuangan, pertanian, NLP, computer vision, dll.)

Permasalahan utama adalah pada domain Perbankan

1. Biaya Operasional dan Sumber Daya Yang terbuang

Promosi digital ini menyebabkan banyak yang tidak tertarik tentang untuk berlangganan deposito berjangka. Hal ini menyebabkan biaya operasional yang tinggi.

2. Kelelahan Nasabah

Terlalu sering atau salah sasaran dalam menghubungi nasabah dapat menyebabkan nasabah merasa terganggu, bahkan berpotensi merusak citra bank.

3. Ketidakimbangan Data

Berdasarkan studi jumlah nasabah yang berlangganan deposito lebih sedikit ketimbang yang berlangganan. Yaitu ('NO') lebih banyak daripada ('YES') hal ini yang menyebabkan tidak keseimbangan data yang berakibat model sulit memprediksi kelas ('yes')

- Manfaat proyek untuk pengguna, bisnis, atau penelitian

- **Pengambilan keputusan berdasarkan data** : Hal ini bisa membuat sebuah peta untuk mempelajari wawasan yang lebih mendalam mengenai informasi karakteristik nasabah yang paling responsif dan juga strategi baru untuk metode pemasaran dan iklan yang tepat dimasa depan.
- **Peningkatan Efisiensi Pemasaran** : Bank dapat mengurangi biaya operasional iklan hingga puluhan persen karena waktu kontak nasabah yang hanya dihabiskan untuk nasabah yang paling mungkin untuk berlangganan.
- **Optimalisasi Berlangganan** : Peningkatan akurasi prediksi secara langsung akan meningkatkan Return on Investment (ROI) dari setiap kampanye pemasaran.

- Studi literatur atau referensi ilmiah (minimal 1–2 sumber wajib)

No	Refrensi	Sumber Terpercaya
1	Prediction of Term Deposit in Bank: Using Logistic Model	Jiang, E., Wang, Z., & Zhao, J. (2022). Prediction of Term Deposit in Bank: Using Logistic Model. <i>BCP Business & Management</i> , 34, 607-614.
2	BankCustomer Decision Prediction on Term Deposit Products Using Random Forest Algorithm on Bank Marketing Campaign Data	Apriadi, E. A., & Bisri, M. (2025). Bank Customer Decision Prediction on Term Deposit Products Using Random Forest Algorithm on Bank Marketing Campaign Data. <i>Journal of Computer Networks, Architecture and High Performance Computing</i> , 7(2), 534-543.

[Jelaskan konteks dan latar belakang proyek]

- Proyek ini berkaitan langsung dengan sebuah bank yang berfokus analisis dan pemodelan data kampanye pemasaran produk deposito berlangganan secara langsung yang dilakukan oleh bank itu. Terdapat sebuah data CSV yang memuat

sejumlah informasi yang berisi demografi, kondisi keuangan, hingga kontak nasabah yang ditawari produk deposito berjangka melalui telepon.

3. BUSINESS UNDERSTANDING / PROBLEM UNDERSTANDING

3.1. Problem Statements

3.1.1. Isu ketidakseimbangan kelas target (Balance Class)

Dataset ini memiliki ketidakseimbangan yang signifikan yang terjadi di kelas target, yaitu signifikan pada target (y) dimana jumlah nasabah yang melakukan deposito berjangka (yes) lebih sedikit dibandingkan yang tidak berlangganan (no). Hal ini membuat tantangan tersendiri untuk membangun sebuah model yang tidak bias terhadap terhadap target (no)

3.1.2. Kurangnya Wawasan prediktif

Bank belum memiliki pemahaman factor factor yang jelas terhadap nasabah (status pekerjaan, Riwayat kredit, dan juga durasi kontak) yang dimana memiliki sebuah korelasi yang bisa dilakukan untuk membuat keputusan berlangganan deposito, sehingga menghambat perumusan strategi yang pemasaran yang optimal

3.1.3. Membutuhkan penangan khusus untuk mengatasi data bias

Hal ini terjadi karena klas target ini memiliki jumlah (no) lebih banyak daripada (yes), hal ini Ketika tidak ditangani secara tepat maka model hanya bisa membaca dan menangkap pola target (no) saja.

3.2. Goals

- Mengembangkan model klasifikasi yang prediktif
- Tidak mengalami bias pada pembacaan pola di model
- Membangun model menggunakan Machine learning dengan target >80% dengan memprediksi variable target
- Mengecek menggunakan 3 model, baseline model, machine learning dan juga deep learning

3.3. Solution Approach

3.3.1. Model 1 - Baseline Model

Alasan : Model 1 memilih Logistic Regression karena dataset ini memiliki kelas target di mana yes atau tidak dan yang berdasarkan bulan, pdays, lama dihubungi dan fitur fitur lain nya (klasifikasi)

3.3.2. Model 2 – Machine Learning

Jawab : Random Forest dipilih karena bisa membangun banyak pohon keputusan yang berbeda dan menggabungkan hasilnya. Ini sangat efektif untuk menangani non linear

3.3.3. Model 3 – Deep Learning

Jawab : Deep learning menggunakan MLPClassifier (Multi-Layer Perceptron Classifier) karena cocok untuk data tabular dan juga karakteristik data yang bersifat klasifikasi. Selain itu Adalah kendala pada saat ketersediaan library di lingkungan eksekusi kode saat ini.

4. Data Understanding

4.1. Informasi Dataset

- Jumlah Baris : 4521
- Jumlah Kolom : 17
- Tipe Data : Data Tabular (Klasifikasi)
- Ukuran Dataset : 451 kb
- Format File : CSV

4.2. Deskripsi Fitur

Nama Fitur	Tipe Data	Deskripsi	Contoh Nilai
Age	Integer	Umur	33, 55, 67
Job	Object	Pekerjaan	Entrepreneur, engineer, Services
Martial	Object	Status Pernikahan	Married, single
Education	Object	Tingkat	Primary,

		Pendidikan	secondary
Default	Object	Apakah memiliki kredit macet atau tidak.	No, Yes
Balance	Integer	Saldo rata-rata tahunan	-333, 295, 30
Housing	Object	Memiliki kredit rumah	Yes, no
Loan	Object	Memiliki pinjaman pribadi	No, yes
Contact	Object	Jenis komunikasi terakhir	Celular, unknown
Day	Integer	Hari kontak terakhir	30,21,4,6,
Month	Object	Bulan kontak terakhir	Jul, May, Aug
Duration	Integer	Durasi Kontak terakhir	324, 123, 231, 12
Campaign	Integer	Jumlah kontak selama kampanye	4, 6, 7, 1
Pdays	Integer	Jumlah hari sejak terakhir kali dihubungi	-1, 211, 249
Previous	Integer	Jumlah kontak sebelum kampanye ini	0, 3, 4, 7, 8

Outcome	Object	Hasil kampanye pemasaran	Unknown, other
Y	Object	Hasil akhir	No, Yes

4.3. Kondisi Data

- Missing Values: Tidak ada (0)
- Duplicate Data: Tidak Ada (0)
- Outliers: Ada [Age, Imbalance, Campaign, dan Balance]
- Imbalanced Data: Ada (Pada fitur Y, Dimana hamper 87% data yang mendominasi adalah no dan sisanya adalah yes)
- Noise: fitur duration dan campaign, karena durasi kontak dan frekuensi panggilan yang sangat tinggi dapat dipengaruhi oleh faktor eksternal seperti kesalahan pencatatan
- Data Quality Issues: Ketidakseimbangan kelas pada variabel target, dan perbedaan nilai skala pada fitur numerik

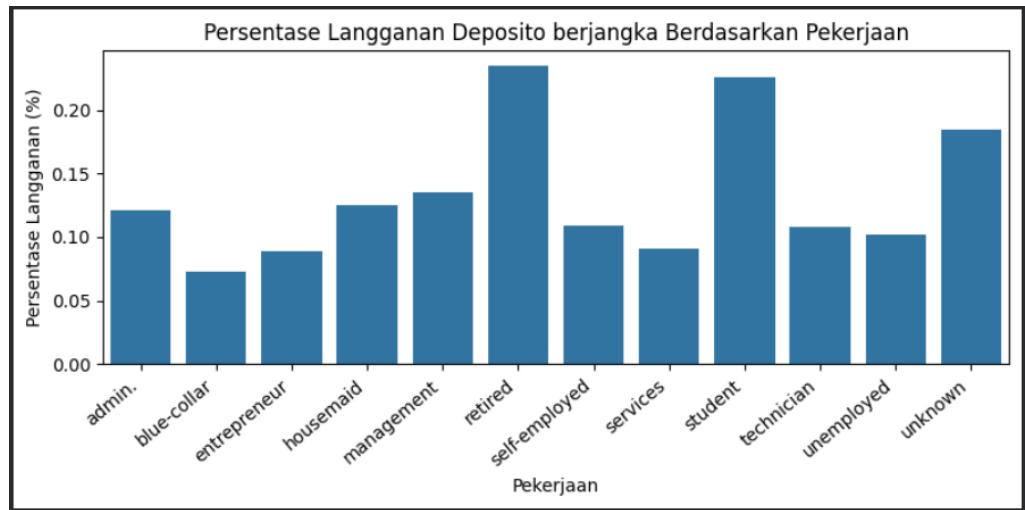
4.4. Exploratory Data Analysis (EDA) - (OPSIONAL)

Requirement : Minimal 3 visualisasi yang bermakna dan insight-nya. Contoh jenis visualisasi yang dapat digunakan

A. Bar Plot

```
# Calculate the proportion of 'yes' for each job
job_y_counts = bank.groupby('job')['y'].value_counts(normalize=True).unstack()
job_y_proportion = job_y_counts['yes'].reset_index()
job_y_proportion.columns = ['job', 'proportion']

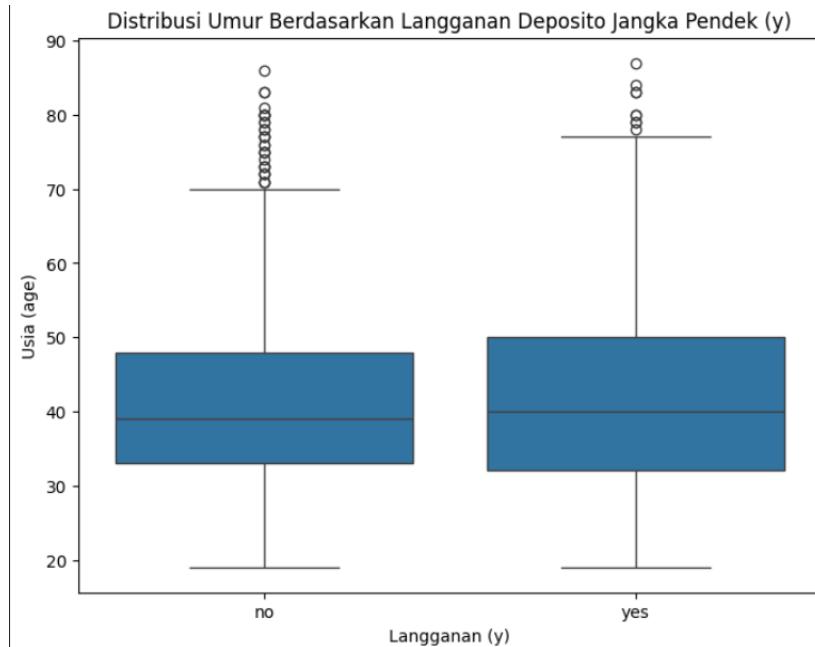
# Visualisasi Data
plt.figure(figsize=(8,4))
sns.barplot(x='job', y='proportion', data=job_y_proportion)
plt.title('Persentase Langganan Deposito berjangka Berdasarkan Pekerjaan')
plt.xlabel('Pekerjaan')
plt.ylabel('Persentase Langganan (%)')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



Deskripsi : Barplot ini memuat sebuah insight dimana persentase langganan deposito berjalan berjangka berdasarkan pekerjaan yang tertinggi adalah **Retired**

B. Box Plot

```
# 1. Bivariate Analysis for Numerical Feature: 'age' vs 'y' (Target)
plt.figure(figsize=(8, 6))
sns.boxplot(x='y', y='age', data=bank)
plt.title('Distribusi Umur Berdasarkan Langganan Deposito Jangka Pendek (y)')
plt.xlabel('Langganan (y)')
plt.ylabel('Usia (age)')
plt.show()
```

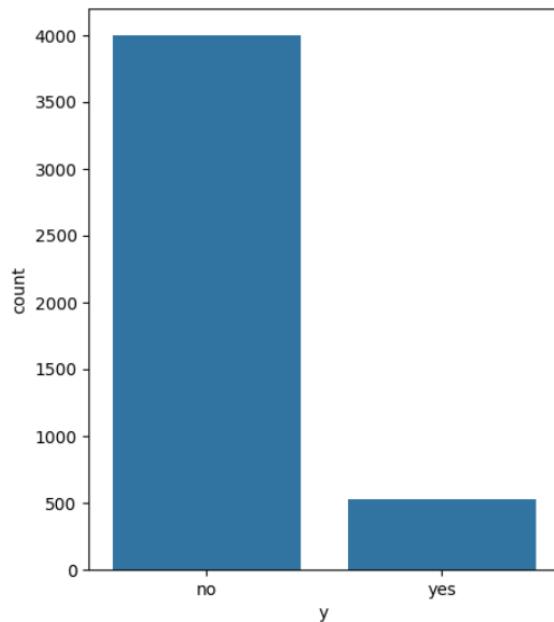


Deskripsi : Distribusi umur adalah nasabah yang memiliki tidak berlangganan (no) maupun berlangganan (yes) memiliki rentang usia yang relatif mirip. Mayoritas nasabah berada pada rentang usia 30–50 tahun.

C. Bar Plot Fitur Y

```
# Bar plot untuk Y

plt.figure(figsize=(5,6))
sns.countplot(x='y', data=bank)
distribusi = bank['y'].value_counts()
```



Deskripsi : Bar plot pada fitur y ini menunjukkan perbandingan yang terdapat di fitur Y, dimana fitur tersebut memuat variable no dan yes yang memiliki ketidakseimbangan jumlah.

5. Data Preparation

5.1. Data Cleaning

- Handling missing values

```
# 5.1 Handling Missing Value

# Handling Missing Value
bank = bank.drop('duration', axis=1)
print("Kolom Duration telah terhapus")

# Handle Special Value -1 in pdays
bank['pdays'] = bank['pdays'].replace(-1, 999)
print("Nilai -1 pada kolom pdays Telah diubah menjadi 999")

... Kolom Duration telah terhapus
Nilai -1 pada kolom pdays Telah diubah menjadi 999
```

Penjelasan : Handling Missing value yaitu menghapus kolom duration, lalu handling dengan Bahasa yang mudah adalah menangani variabel -1 pada kolom pdays. Hal ini digantikan dengan 999 yang berarti itu sudah lama tidak dihubungi.

- Removing duplicates

```
▶ # 5.2 Removing Duplicates

cekDuplikat = len(bank)
bank.drop_duplicates(inplace=True)
cekSetelahDrop = len(bank)
print(f"Jumlah Duplikat Setelah Dihapus: ", {cekDuplikat - cekSetelahDrop})

... Jumlah Duplikat Setelah Dihapus:  {0}
```

Penjelasan : Menghapus duplikat yang ada, Langkah pertama ini merupakan cek duplikat dimana akan menghitung semua dari jumlah dataset setelah di cek maka akan dilakukan drop dengan parameter (inplace=True)

- Handling outliers

```
▶ # 5.3 Handling Outliers

numerik_fitur = ['balance', 'campaign']

for col in numerik_fitur:
    upper_limit = bank[col].quantile(0.95)
    bank[col] = np.where(bank[col] > upper_limit, upper_limit, bank[col])
    print(f"Outliers pada kolom {col} telah dibatasi pada Quantile 92 ({upper_limit:.2f}).")

... Outliers pada kolom balance telah dibatasi pada Quantile 92 (6102.00).
Outliers pada kolom campaign telah dibatasi pada Quantile 92 (8.00).
```

Penjelasan : Disini saya menggunakan Teknik **Capping (Winsorization)**, Teknik tersebut adalah Teknik yang menangani nilai ekstrim pada kumpulan data dengan cara membatasi nilai pada ambang batas tertentu. Disini membatasi dengan quantile 92 pada kolom balance dan kolom campaign

- Data type conversion

```
▶ # 5.4 Data Conversation

bank['y'] = bank['y'].map({'yes': 1, 'no': 0})
print("Variabel Target y telah dikonversi menjadi bilangan binner (1=yes, 0=no )")

# Tampilkan setelah dilakukan cleaning
print("\nInformasi Data Setelah Data Cleaning ")
print(bank.head().to_markdown(index=False, numalign="left", stralign="left"))

... Variabel Target y telah dikonversi menjadi bilangan binner (1=yes, 0=no )
```

Informasi Data Setelah Data Cleaning																
age	job	marital	education	default	balance	housing	loan	contact	day	month	campaign	pdays	previous	poutcome	y	
30	unemployed	married	primary	no	1787	no	no	cellular	19	oct	1	999	0	unknown	0	
33	services	married	secondary	no	4789	yes	yes	cellular	11	may	1	339	4	failure	0	
35	management	single	tertiary	no	1350	yes	no	cellular	16	apr	1	330	1	failure	0	
30	management	married	tertiary	no	1476	yes	yes	unknown	3	jun	4	999	0	unknown	0	
59	blue-collar	married	secondary	no	0	yes	no	unknown	5	may	1	999	0	unknown	0	

Penjelasan : Data conversation ini melakukan pengubahan pada kolom Y dimana no:0 dan yes menjadi 1

5.2. Feature Engineering

- Creating New Feature

```
▶ # 1. Creating new Feature

bank['previously_contact'] = np.where(bank['pdays'] != 999, 1, 0)
print("Kolom previously_contact telah dibuat ")

# Create negativeve Balance
bank['negative_balanced'] = np.where(bank['balance'] < 0, 1, 0)
print("Negfative Kolom Berhasil Dibuat")

# Create Category Group in pdays
def categorize_pdays(pdays):
    if pdays == 999:
        return 'Not Contacted'
    elif pdays < 30:
        return 'Fresh Contacted'
    elif pdays < 90:
        return 'Regular Contacted'
    else:
        return 'Old Contacted'

bank['pdays_group'] = bank['pdays'].apply(categorize_pdays)
print("Kolom pdays telah dibuat")

...
... Kolom previously_contact telah dibuat
... Negfative Kolom Berhasil Dibuat
... Kolom pdays telah dibuat
```

Penjelasan : pada kode ini membuat sebuah fitur baru yang Bernama previously_contact dan negative balance. Ini bertujuan untuk mendefinisikan dan mengelompokkan grup pada menjadi 4 bagian (not contacted, fresh contacted, regular contacted, dan old contacted).

- Feature Extraction

```
▶ # Feature Extraction

month_mapping = [
    'jan': 1, 'feb': 2, 'mar': 3, 'apr': 4,
    'may': 5, 'jun': 6, 'jul': 7, 'aug': 8,
    'sep': 9, 'oct': 10, 'nov': 11, 'dec': 12]

bank['month_num'] = bank['month'].map(month_mapping)
print("Fitur Month Number berhasil Di ekstrak")

...
... Fitur Month Number berhasil Di ekstrak
```

Penjelasan : Fitur extraction ini merupakan fitur dimana nama nama bulan diekstrak menjadi number sesuai urutan bulan dalam setahun

- **Feature selection**

```
# 5.3 Feature Selection

# Drop TO column predictive feature

columns_to_drop = ['day', 'poutcome', 'month']
existing_columns_to_drop = [col for col in columns_to_drop if col in bank.columns]

if existing_columns_to_drop:
    bank = bank.drop(existing_columns_to_drop, axis=1)
    print(f"Kolom {existing_columns_to_drop} telah terhapus")
else:
    print("Tidak ada kolom yang perlu dihapus dari daftar yang ditentukan karena semuanya sudah tidak ada.")

# Display the head of the engineering data
print("\nData setelah Feature Engineering")
print(bank.head().to_markdown(index=False, numalign='left', stralign = "left"))

... Kolom ['day', 'poutcome', 'month'] telah terhapus
```

Penjelasan : Melakukan penghapusan fitur yang tidak relevan pada tujuan pembuatan model dan menghapus fitur yang dianggap kurang prediktif.

- **Dimensionality Reduction**

```
# Dimensionality Reduction

X = bank.drop('y', axis=1)
y = bank['y']

X_encoded = pd.get_dummies(bank, drop_first=True)

from sklearn.decomposition import PCA

X_train_scaled = preprocessor.fit_transform(X_train)
X_test_scaled = preprocessor.transform(X_test)
```

```
pca = PCA(n_components=0.95, random_state=42)

X_train_pca = pca.fit_transform(X_train_scaled)
X_test_pca = pca.transform(X_test_scaled)

print("Fitur sebelum PCA:", X_train.shape[1])
print("Fitur setelah PCA:", X_train_pca.shape[1])

Fitur sebelum PCA: 16
Fitur setelah PCA: 17
```

Penjelasan : Untuk Langkah yang pertama ini adalah melakukan pemisahan fitur dan target. Melakukan tahap kedua adalah scaler yang berfungsi sebagai mengubah skala fitur numerik agar memiliki rentang nilai

yang serupa. Untuk yang terakhir melakukan tahap PCA yang berfungsi untuk mengurangi jumlah variabel dan tetap mempertahankan informasi penting.

5.3. Data Transformation

```
# 1. Mendefinisikan Kelompok Fitur

# Fitur numerik
numerik_fitur = ['age', 'pdays', 'campaign', 'balance', 'previous', 'month_num']

# Fitur Ordinal
ordinal_fitur = ['default', 'housing', 'loan', 'education', 'pdays_group']

# Mendefinisikan untuk kategori ordinal encoding
education_mapping = ['unknown', 'primary', 'secondary','tertiary']
pdays_group_mapping = ['Not Contacted', 'Fresh Contacted', 'Regular Contacted', 'Old Contacted']

# Fitur Kategori
kategori_fitur = ['job', 'marital', 'contact']
```

Penjelasan : Kode ini memuat mendefinisikan kelompok fitur dengan nilai kontinu numerik yang dapat digunakan langsung dalam perhitungan sistematis machine learning. Untuk yang kedua adalah mengelompokkan fitur ordinal yang memiliki tingkatan nilai sehingga tidak tepat jika dilakukan one hot encoding. Kemudian melakukan pendefinisian urutan kategori yang akan digunakan dalam ordinal mapping. Dan yang terakhir adalah kelompok yang tidak memiliki urutan antar kategori

5.4. Data Splitting

A. Split data

```
X = bank.drop('y', axis=1)
y = bank['y']

from sklearn.model_selection import train_test_split

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

Penjelasan : Tahap ini merupakan pemisahan tahap dimana memisahkan antara fitur dan target, kemudian melakukan pemisahan data yang dimana 20% untuk data uji dan 80% untuk data latih. Random state yang yang digunakan 42 ini menjamin konsistensi data dan dapat direproduksi. Yang

terpenting setiap kali melakukan pemisahan data hasil split nya akan tetap sama.

B. Implementasi PCA

```
from sklearn.decomposition import PCA

X_train_scaled = preprocessor.fit_transform(X_train)
X_test_scaled = preprocessor.transform(X_test)

pca = PCA(n_components=0.95, random_state=42)

X_train_pca = pca.fit_transform(X_train_scaled)
X_test_pca = pca.transform(X_test_scaled)

print("Fitur sebelum PCA:", X_train.shape[1])
print("Fitur setelah PCA:", X_train_pca.shape[1])

...
Fitur sebelum PCA: 16
Fitur setelah PCA: 17

num_cols = X.select_dtypes(include=['int64', 'float64']).columns
cat_cols = X.select_dtypes(include=['object']).columns

preprocessor = ColumnTransformer([
    ('num', StandardScaler(), num_cols),
    ('cat', OneHotEncoder(handle_unknown='ignore', sparse_output=False), cat_cols)
])
```

Penjelasan :

- Data dipisahkan menjadi fitur dan target yang dimana variable target mendefinisikan hasil dari klasifikasi.
- Langkah selanjutnya adalah Langkah split training dengan 80% untuk data latih dan 20% untuk data uji dan menggunakan Teknik stratified sampling untuk menjaga agar tetap proporsional pada kelas target.
- PCA hanya dilatih pada data training dan PCA memiliki jumlah komponen yang otomatis. Hal ini dilakukan pada setelah split data training dan kemudian diterapkan pada data latih yang bertujuan untuk mengatasi data Leakage
- Preprocessing dilatih hanya pada data training dan kemudian diterapkan pada data testing untuk mencegah terjadinya data leakage.

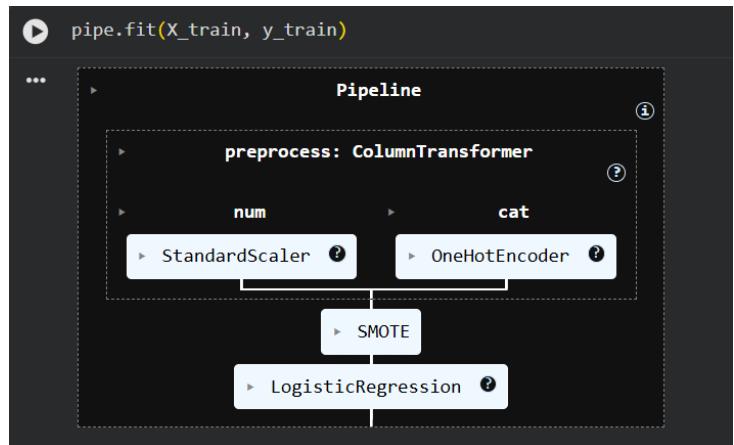
- Langkah terakhir adalah melakukan Standardscaler yang berfungsi untuk menampilkan fitur numerik dan OneHotEncoder digunakan untuk mengubah fitur ordinal yang akan mempresentasikan numerik tanpa menyebabkan error pada kategori yang tidak dikenal

5.5. Data Balancing (Jika Diperlukan)

- A. Untuk data balancing pada kasus dataset bank ini saya menggunakan Teknik SMOTE. Dimana Teknik Smote ini adalah Teknik yang digunakan dalam machine learning untuk menangani ketidakseimbangan data dengan cara membuat sampel data sintetis (buatan) dari kelas minoritas

```
# Base model
log_reg = LogisticRegression(max_iter=5000, solver='lbfgs')

pipe = ImbPipeline([
    ('preprocess', preprocess),
    ('smote', SMOTE(random_state=42)),
    ('logreg', LogisticRegression(
        solver='lbfgs',
        max_iter=5000
    ))
])
```



```

y_pred = pipe.predict(X_test)
y_proba = pipe.predict_proba(X_test)[:,1]

print(classification_report(y_test, y_pred))
print("ROC AUC:", roc_auc_score(y_test, y_proba))

...      precision    recall   f1-score   support
          0       0.93     0.64     0.76     801
          1       0.19     0.63     0.29     104

accuracy                           0.64     905
macro avg       0.56     0.64     0.52     905
weighted avg    0.85     0.64     0.71     905

ROC AUC: 0.6959329684048785
  
```

```

param_grid = {
    'logreg_C': [0.001, 0.01, 0.1, 1, 10],
    'logreg_penalty': ['l2'],
    'logreg_class_weight': [None, 'balanced'],
}

cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

grid = GridSearchCV(pipe, param_grid, cv=cv, scoring='roc_auc', n_jobs=-1)
grid.fit(X_train, y_train)

print("Best params:", grid.best_params_)
best_model = grid.best_estimator_

... Best params: {'logreg_C': 0.01, 'logreg_class_weight': None, 'logreg_penalty': 'l2'}
  
```

Penjelasan : Setelah melakukan Teknik Smote maka akan digunakan pipeline untuk mengumpulkan keseluruhan data yang sudah di StandardScaler dan OneHotCoding. Maka setelah itu dilakukan pengecekan awal yang berisi untuk melihat ROC-AUC.

5.6. Ringkasan Data Preparation

1. Apa yang dilakukan ?

Pada tahap ini melakukan sebuah data cleaning yang bertujuan untuk agar isi dari data benar benar bersih dan tidak ada yang duplikat atau kosong. Kemudian melakukan handing value missing, removing duplicate, dan data conversation. Pada tahap data preparation semua yang dilakukan adalah penitig karena untuk memastikan Ketika dilakukan pelatihan model benar benar bisa menangkap target yang dituju. Menambahkan fitur engineering yang berfungsi sebagai membuat fitur baru dan juga mengekstrak fitur dari kolom yang sudah ada, sebagai contoh adalah fitur ekstrak pada kolom month yang awalnya teks diubah menjadi angka sesuai urutan bulan dalam setahun

2. Mengapa penting

Karena untuk memastikan bahwa Ketika dilakukan pemodelan ini tidak bias terhadap kolom target yaitu no, hal ini sangat penting dilakukan mengingat variable no yang ada di dalam kolom y itu sangat mendominasi.

3. Bagaimana implementasinya

Dengan menggunakan Teknik SMOTE pada hyperparameter mengatasi masalah data tidak seimbang (imbalanced data) dalam machine learning dengan cara membuat sampel data sintetis (buatan) dari kelas minoritas, sehingga jumlah data kelas minoritas bertambah dan proporsi dataset menjadi lebih seimbang tanpa harus menggandakan data asli secara langsung, yang meningkatkan performa model klasifikasi.

6. Modeling

6.1. Baseline Model

6.1.1. Deskripsi Model

Nama Model : Logistic Regression

Teori Singkat : Sebuah model yang memperkirakan kemungkinan suatu peristiwa terjadi, seperti memilih atau tidak memilih, berdasarkan dataset variabel independen yang diberikan.

6.1.2. Hyperparameter

```
# Hyperparameter

params = [
    "solver": ["liblinear", "lbfgs"],
    "C": [0.01, 0.1, 1, 10],
    "class_weight": [None, "balanced"],
    "max_iter": [200, 500]
]
```

Penjelasan :

- Solver = Menentukan algoritma optimisasi yang digunakan untuk mencari parameter terbaik (koefisien).
- (Regularization Strength) = Mengontrol regulasi dan menghindari terjadinya overfitting
- ClassWeight = Mengatur bobot kelas saat training
- Max_iter = Menentukan jumlah iterasi maksimum saat optimisasi.

6.1.3. Implementasi

```
# --- MODEL ---

y_pred = best_model.predict(X_test)
y_proba = best_model.predict_proba(X_test)[:,1]

report = classification_report(y_test, y_pred, output_dict=True)
Auc = roc_auc_score(y_test, y_proba)
```

Penjelasan : Ini adalah kode implementasi kode untuk melakukan pemubatan model logistic Regression

6.1.4. Hasil Awal

```

# Hasil Awal
print("Hasil model logistiv regression")

# Define LR metrics
acc_lr = report["accuracy"]
auc_lr = Auc
f1_lr = report["1"]["f1-score"] # F1-score for class 1 (yes)

result = {
    "Logistic Regression" : {
        "Accuracy" : round(acc_lr, 4),
        "AUC" : round(auc_lr, 4),
        "Recall_1" : round(report["1"]["recall"], 4)
    }
}

pd.DataFrame(result).T

```

... Hasil model logistiv regression

	Accuracy	AUC	Recall_1
Logistic Regression	0.6508	0.6998	0.6538

Penjelasan : mendapatkan accuracy sebesar 0.65 dengan AUC 0,69 dan Recall 0,65. Hal ini menunjukkan bahwa model memiliki akurasi yang cukup tetapi sekitar 65% bisa mengenali kelas positif dan model memiliki kemampuan sebesar 69% untuk mengenali target NO dan Yes Secara keseluruhan.

6.2. Machine learning / Advanced

6.2.1. Deskripsi Model

Nama Model : Random Forest

Teori Singkat : Sebuah model algoritma machine learning yang digunakan untuk melakukan klasifikasi dan regresi dengan cara memecah dataset menjadi subset yang lebih kecil berdasarkan fitur tertentu

6.2.2. Hyperparameter

```
# Model

rf = RandomForestClassifier(
    n_estimators = 200,
    max_depth = 5,
    min_samples_split = 2,
    min_samples_leaf = 1,
    max_features = 'sqrt',
    random_state = 42,
    class_weight = 'balanced',
    n_jobs = -1
)
```

Penjelasan :

- n_estimators = Jumlah pohon dalam assemble
- maxdepth = kedalaman ,maksimum setiap pohon untuk mencegah overfitting
- min_sample_split = Minimun sample untuk pemisahan node
- min_sample_leaf = Minimum sample untuk leaf node
- max_features = fitur yang untuk setiap split agar lebih stabil
- class_weight = Menangai Imbalanced Data
- Random_state = Reproducibility, agar setiap pembagian data latih dan data uji untuk mengontrol angka acak.
- Class_weight = Digunakan untuk menangani data yang tidak seimbang
- N_jobs = Menggunakan seluruh CPU core

6.2.3. Implementasi

```

▶ # Identifikasi kolom kategorikal
categorical_cols = X.select_dtypes(include=['object']).columns.tolist()

# Model

rf = RandomForestClassifier(
    n_estimators = 200,
    max_depth = 5,
    min_samples_split = 2,
    min_samples_leaf = 1,
    max_features = 'sqrt',
    random_state = 42,
    class_weight = 'balanced',
    n_jobs = -1
)

model = Pipeline([
    ("preprocess", preprocess),
    ("rf", rf)
])

model.fit(X_train, y_train)

```

6.2.4. Hasil Awal

```

▶ # Clasifikasi Report

hasilReport = classification_report(y_test, y_pred)
print(hasilReport)

...          precision    recall   f1-score   support
          0       0.93      0.79      0.86      801
          1       0.26      0.56      0.35      104
          accuracy           0.76      905
          macro avg       0.59      0.67      0.60      905
          weighted avg     0.85      0.76      0.80      905

```

Untuk Akurasi pada model Random forest ini 0.76 hal ini menunjukkan model memiliki nilai yang bagus untuk perolehan dengan F1_score adalah 35% dapat membaca kelas positif yang meningkat daripada logistic regression dan 69% bisa mengenali keseluruhan kelas target yaitu No dan YES

6.3. Model Deep Learning

6.3.1. Deskripsi Model

Nama Model : MLP (Multi-Layer Perceptron)

Alasan : Karena data ini bersifat tabular dan klasifikasi yang memiliki kelas target yaitu No dan Yes berdasarkan fitur fitur lainnya.

Teori Singkat : serangkaian lapisan tersembunyi tak terbatas yang terletak di antara lapisan keluaran dan lapisan masukan . Data melewati jalur maju dari lapisan masukan ke lapisan keluaran dalam MLP, setara dengan jaringan feed-forward. Teknik pembelajaran backpropagation digunakan untuk melatih semua node dalam MLP.

6.3.2. Arsitektur Model

No	Layer	Jumlah Neuron	Aktivasi	Keterangan
1	Input Layer	input_dim	-	Menerima data fitur hasil preprocessing
2	Dense (Hidden 1)	128	ReLU	Menangkap pola non-linear kompleks
3	Dropout	0,3	-	Mencegah overfitting
4	Dense (Hidden 2)	64	ReLU	Ekstraksi fitur tingkat lanjut
5	Dropout	0,3	-	Regularisasi
6	Dense (Output)	1	Sigmoid	Prediksi probabilitas kelas biner

6.3.3. Input & Preprocessing Khusus

Terdapat sebuah input shape dimana input_dim merupakan jumlah fitur setelah proses preprocessing (scaling, encoding, dan/atau dimensionality reduction).

Preprocessing Khusus untuk Deep Learning ini Preprocessing dilakukan sebelum model deep learning dilatih untuk memastikan data berada dalam format numerik dan skala yang sesuai.

1. Standarisasi Fitur Numerik
2. Encoding Fitur kategorikal
3. Dimensionality Reduction (Opsional)

6.3.4. Hyperparameter

Hyperparameter	Nilai	Keterangan
Optimizer	Adam	Optimisasi adaptif
Learning Rate	0.001 (default)	Stabil & cepat konvergen
Loss Function	Binary Crossentropy	Untuk klasifikasi biner
Metrics	Accuracy, AUC	Evaluasi performa & imbalance
Batch Size	32	Trade-off kecepatan & stabilitas
Epochs (Max)	10	Training maksimum
Validation Split	0.2	Data Validasi
Callback	EarlyStopping	Mencegah overfitting
EarlyStopping Monitor	val_loss	Memantau performa validasi

6.3.5. Implementasi

```

▶ # Implementasi

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers

# The input dimension should be based on the processed data
# input_dim = X_train.shape[1] # This would be incorrect for unprocessed X_train
input_dim = X_train_processed.shape[1] # Use the shape of the processed data

# Early Stopping
early_stopping = keras.callbacks.EarlyStopping(
    monitor='val_loss',
    patience=5,
    restore_best_weights=True
)

# Model MPL
model_deep_learning = keras.Sequential([
    layers.Dense(128, activation='relu', input_shape=(input_dim,)),
    layers.Dropout(0.3),
    layers.Dense(64, activation='relu'),
    layers.Dropout(0.3),
    layers.Dense(1, activation='sigmoid')
])

```



```

▶ # Compile Model
model_deep_learning.compile(
    optimizer='adam',
    loss='binary_crossentropy',
    metrics=['accuracy', keras.metrics.AUC(name='auc')]
)

# Training
history = model_deep_learning.fit(
    X_train_processed, y_train, # Use the preprocessed training data
    epochs = 10,
    batch_size=32,
    validation_split=0.2,
    callbacks=[early_stopping],
    verbose=1
)

```



```

*** /usr/local/lib/python3.12/dist-packages/keras/src/layers/core/dense.py:93: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using
super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Epoch 1/10
91/91      2s 8ms/step - accuracy: 0.8739 - auc: 0.5389 - loss: 0.4058 - val_accuracy: 0.8950 - val_auc: 0.6921 - val_loss: 0.3107
Epoch 2/10
91/91      0s 4ms/step - accuracy: 0.8884 - auc: 0.6015 - loss: 0.3541 - val_accuracy: 0.8950 - val_auc: 0.7180 - val_loss: 0.3154
Epoch 3/10
91/91      0s 4ms/step - accuracy: 0.8776 - auc: 0.7114 - loss: 0.3400 - val_accuracy: 0.8950 - val_auc: 0.7246 - val_loss: 0.3057
Epoch 4/10
91/91      0s 4ms/step - accuracy: 0.8848 - auc: 0.6891 - loss: 0.3294 - val_accuracy: 0.8881 - val_auc: 0.7356 - val_loss: 0.3013
Epoch 5/10
91/91      0s 4ms/step - accuracy: 0.8834 - auc: 0.7104 - loss: 0.3264 - val_accuracy: 0.8923 - val_auc: 0.7359 - val_loss: 0.3043
Epoch 6/10
91/91      0s 5ms/step - accuracy: 0.8801 - auc: 0.7252 - loss: 0.3380 - val_accuracy: 0.8923 - val_auc: 0.7372 - val_loss: 0.3021
Epoch 7/10
91/91      1s 7ms/step - accuracy: 0.8801 - auc: 0.7163 - loss: 0.3355 - val_accuracy: 0.8936 - val_auc: 0.7413 - val_loss: 0.3020
Epoch 8/10
91/91      1s 7ms/step - accuracy: 0.8879 - auc: 0.7247 - loss: 0.3169 - val_accuracy: 0.8936 - val_auc: 0.7334 - val_loss: 0.3046
Epoch 9/10
91/91      1s 6ms/step - accuracy: 0.8703 - auc: 0.7472 - loss: 0.3388 - val_accuracy: 0.8936 - val_auc: 0.7242 - val_loss: 0.3047

```

6.3.6. Training Proses

- A. Training Proses memakan waktu 30 detik
- B. Training Dan Visalization

```

# Training loss Visualization

history_get = history.history

plt.figure(figsize=(12,4))

plt.subplot(1,2,1)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Training & Validation Loss')
plt.legend()

```



Penjelasan :

Training Loss:

- Menurun secara konsisten dari ~0.374 (epoch 0) ke ~0.317 (epoch 8)
- Penurunan tajam di awal (epoch 0-2), kemudian lebih gradual
- Menunjukkan model terus belajar dari data training

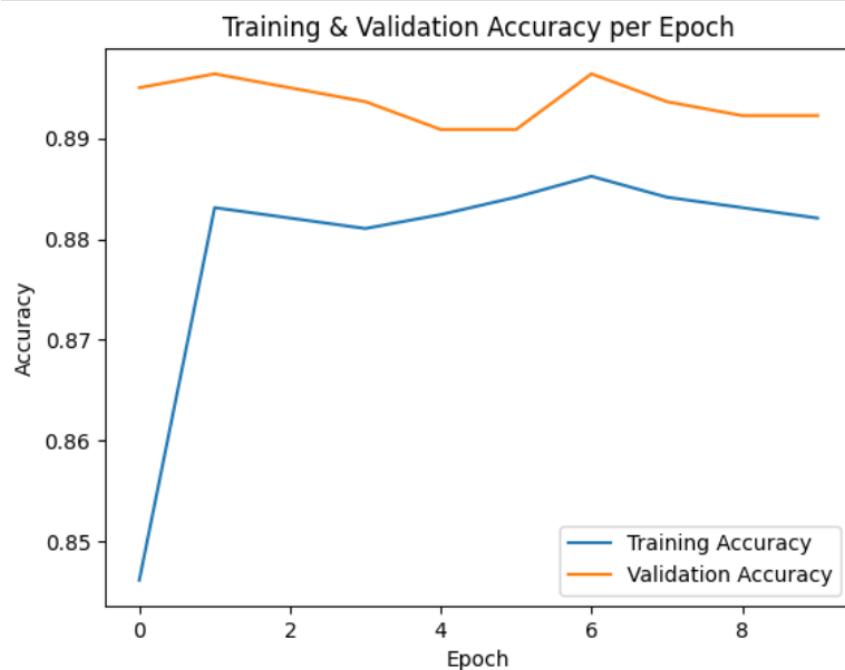
Validation

- Dimulai dari ~0.310 (epoch 0)
- Naik sedikit di epoch 1 (~0.313)
- Menurun hingga epoch 3 (~0.302)
- Stabil/plateau di sekitar 0.302-0.306 dari epoch 3-8

Kesimpulan : Tidak ada tanda overfitting yang signifikan (validation loss tidak naik drastis) Gap antara train dan validation loss relatif kecil (~0.01-0.02)

C. Training & Validation Accuracy/Metric per epoch

```
▶ # Plot Accuracy
    plt.figure()
    plt.plot(history.history['accuracy'], label='Training Accuracy')
    plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.title('Training & Validation Accuracy per Epoch')
    plt.legend()
    plt.show()
```



Penjelasan :

Training Accuracy

- Melonjak tajam dari ~0.846 (epoch 0) ke ~0.883 (epoch 1)
- Naik perlahan hingga puncak ~0.886 di epoch 6

- Sedikit turun dan stabil di ~0.882 (epoch 8-9)
- Fluktuasi kecil setelah epoch 1

Validation Accuracy

- Dimulai tinggi di ~0.895 (epoch 0)
- Naik ke puncak ~0.897 (epoch 1)
- Turun ke ~0.890 (epoch 3-4)
- Naik lagi ke ~0.896 (epoch 6)
- Turun dan stabil di ~0.892 (epoch 8-9)

Kesimpulan : **Akurasi ~89%** (cukup baik, tergantung problem) dan tidak mengalami **overfitting** secara signifikan. Untuk Performa sudah stabil setelah epoch 3-4

6.3.7. Model summary

```
# Model Summary
model_deep_learning.summary()

... Model: "sequential"
+-----+-----+-----+
| Layer (type) | Output Shape | Param # |
+-----+-----+-----+
| dense (Dense) | (None, 128) | 5,248   |
| dropout (Dropout) | (None, 128) | 0        |
| dense_1 (Dense) | (None, 64) | 8,256   |
| dropout_1 (Dropout) | (None, 64) | 0        |
| dense_2 (Dense) | (None, 1) | 65      |
+-----+-----+-----+
Total params: 40,709 (159.02 KB)
Trainable params: 13,569 (53.00 KB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 27,140 (106.02 KB)
```

7. Evaluation

7.1.1. Model Baseline

```

▶ y_pred_lr = best_model.predict(X_test)
y_proba_lr = best_model.predict_proba(X_test)[:,1]

print(classification_report(y_test, y_pred_lr))
print("ROC AUC:", roc_auc_score(y_test, y_proba_lr))

...
precision    recall   f1-score   support
...
0            0.94      0.65      0.77      801
1            0.20      0.65      0.30      104
...
accuracy          0.65
macro avg       0.57      0.65      0.53      905
weighted avg     0.85      0.65      0.71      905

ROC AUC: 0.6997863247863249

```

- Hasil Evaluasi ini Recall (Kelas 'yes', 0.65): Model ini memiliki Recall terbaik untuk kelas minoritas. Ini berarti ia paling berhasil menangkap pelanggan potensial ('yes')
- Kelemahan: Akurasinya rendah 65.4% dan Precision-nya sangat rendah. Model ini terlalu "agresif" dan sering memprediksi 'yes' meskipun hasilnya salah (banyak False Positives)

7.1.2. Model Advanced (Machine Learning)

```

▶ y_pred_rf = model_rf.predict(X_test_processed)
y_proba_rf = rf.predict_proba(X_test_processed)[:,1]

print(classification_report(y_test, y_pred_rf))
print("ROC-AUC: ", roc_auc_score(y_test, y_proba_rf))

...
precision    recall   f1-score   support
...
0            0.93      0.79      0.86      801
1            0.26      0.56      0.35      104
...
accuracy          0.76
macro avg       0.59      0.67      0.60      905
weighted avg     0.85      0.76      0.80      905

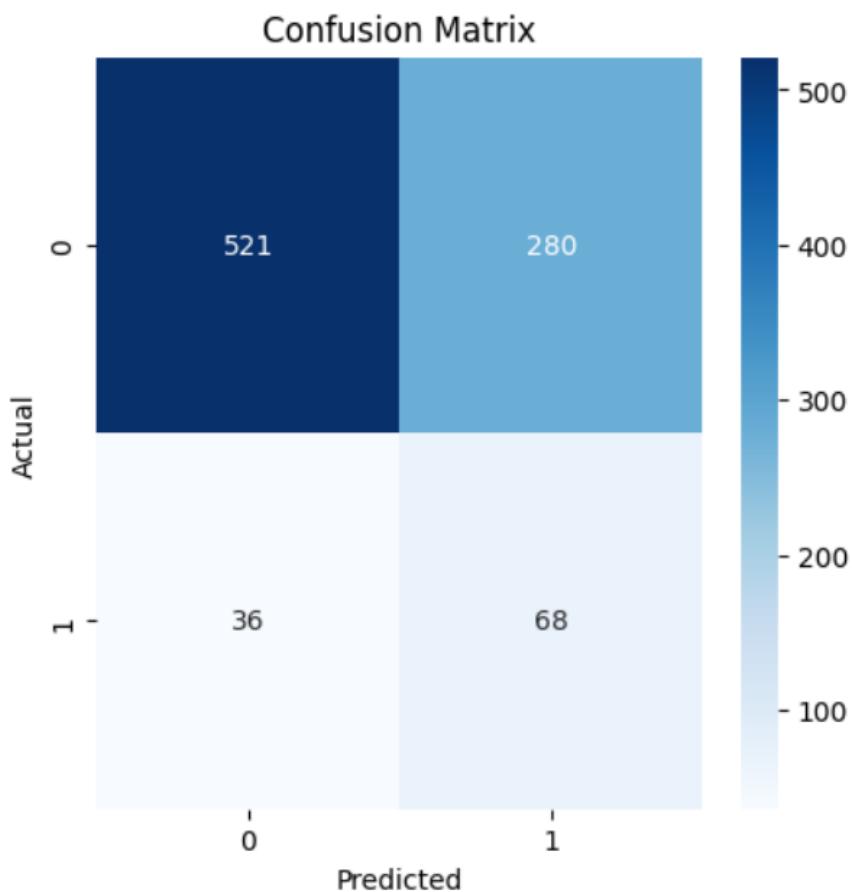
ROC-AUC: 0.7385119562085853

```

- Confusion Matrix

```
# Confusion Matrix

conf_matrix = confusion_matrix(y_test, y_pred_lr)
plt.figure(figsize=(5,5))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
```



Penjelasan :

- Precision: 0.93 - Dari semua prediksi Class 0, 93% benar
- Recall: 0.79 - Dari semua data Class 0 yang sebenarnya, hanya 79% yang terdeteksi
- F1-Score: 0.86 - Keseimbangan precision-recall cukup baik
- Strength: Precision tinggi (sedikit false positive)

- Feature Importance (jika applicable)

Ya Sangat Disarankan untuk Feature Importance karena model ini secara inheren mampu mengukur kontribusi masing-masing fitur terhadap proses pengambilan keputusan. Analisis ini memberikan interpretasi tambahan terhadap hasil klasifikasi

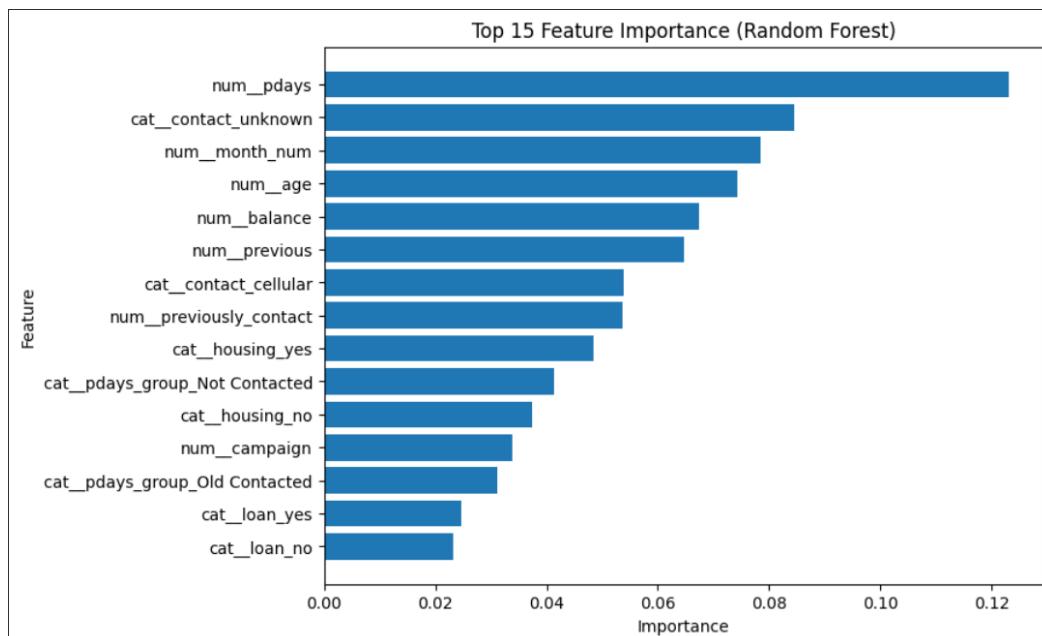
```
# Ambil feature importance
importances = rf.feature_importances_

# Ambil nama fitur setelah preprocessing
feature_names = preprocessor.get_feature_names_out()

# Buat dataframe
feat_imp = pd.DataFrame({
    'feature': feature_names,
    'importance': importances
}).sort_values(by='importance', ascending=False)

# Ambil top 15 fitur terpenting
top_features = feat_imp.head(15)

# Plot
plt.figure(figsize=(8, 6))
plt.barh(top_features['feature'], top_features['importance'])
plt.xlabel('Importance')
plt.ylabel('Feature')
plt.title('Top 15 Feature Importance (Random Forest)')
plt.gca().invert_yaxis()
plt.show()
```



7.1.3. Model Deep learning

```

y_proba_deepLearning = model_deep_learning.predict(X_test_processed).ravel()
y_pred_deepLearning = (y_proba_deepLearning >= 0.3).astype(int)

report_dl = classification_report(y_test, y_pred_deepLearning, output_dict=True)
acc_dl = report_dl["accuracy"]
f1_dl = report_dl["1"]["f1-score"] # F1-score for class 1 (yes)
auc_dl = roc_auc_score(y_test, y_proba_deepLearning)

print(classification_report(y_test, y_pred_deepLearning))
print("ROC-AUC: ", auc_dl)

...

```

29/29 0s 2ms/step

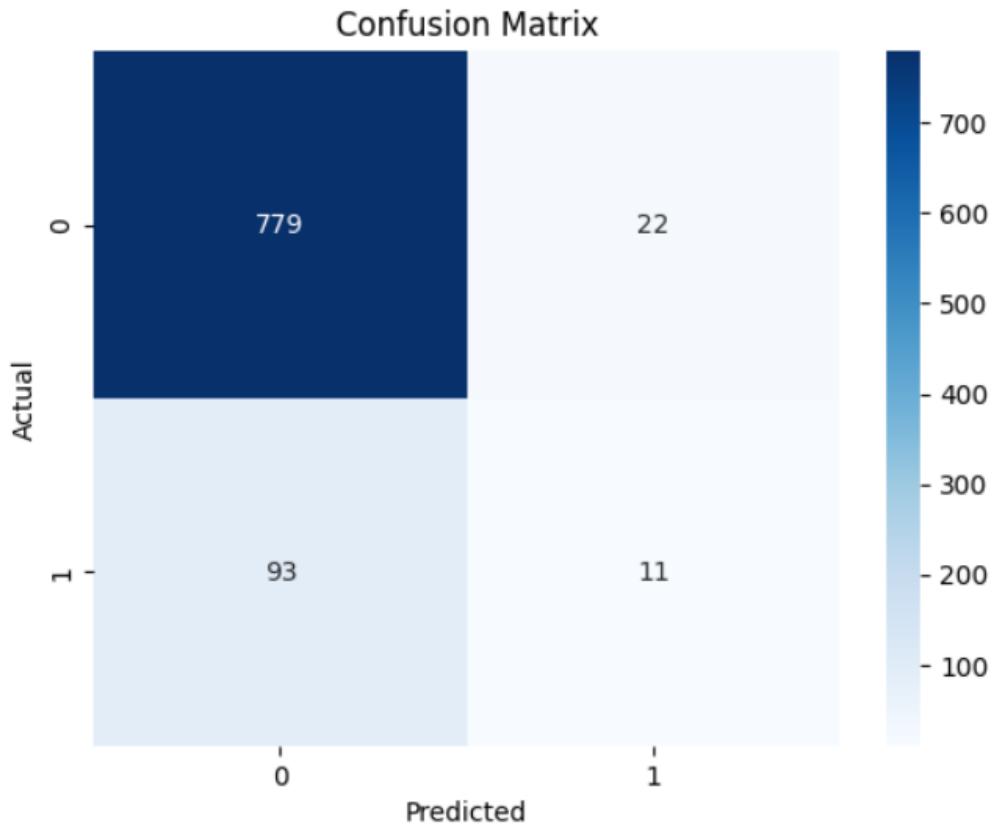
	precision	recall	f1-score	support
0	0.89	0.97	0.93	801
1	0.33	0.11	0.16	104
accuracy			0.87	905
macro avg	0.61	0.54	0.55	905
weighted avg	0.83	0.87	0.84	905

ROC-AUC: 0.7442979928934985

```

confMatrix_DeepLearning = confusion_matrix(y_test, y_pred_deepLearning)
sns.heatmap(confMatrix_DeepLearning, annot=True, fmt='d', cmap='Blues')
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()

```

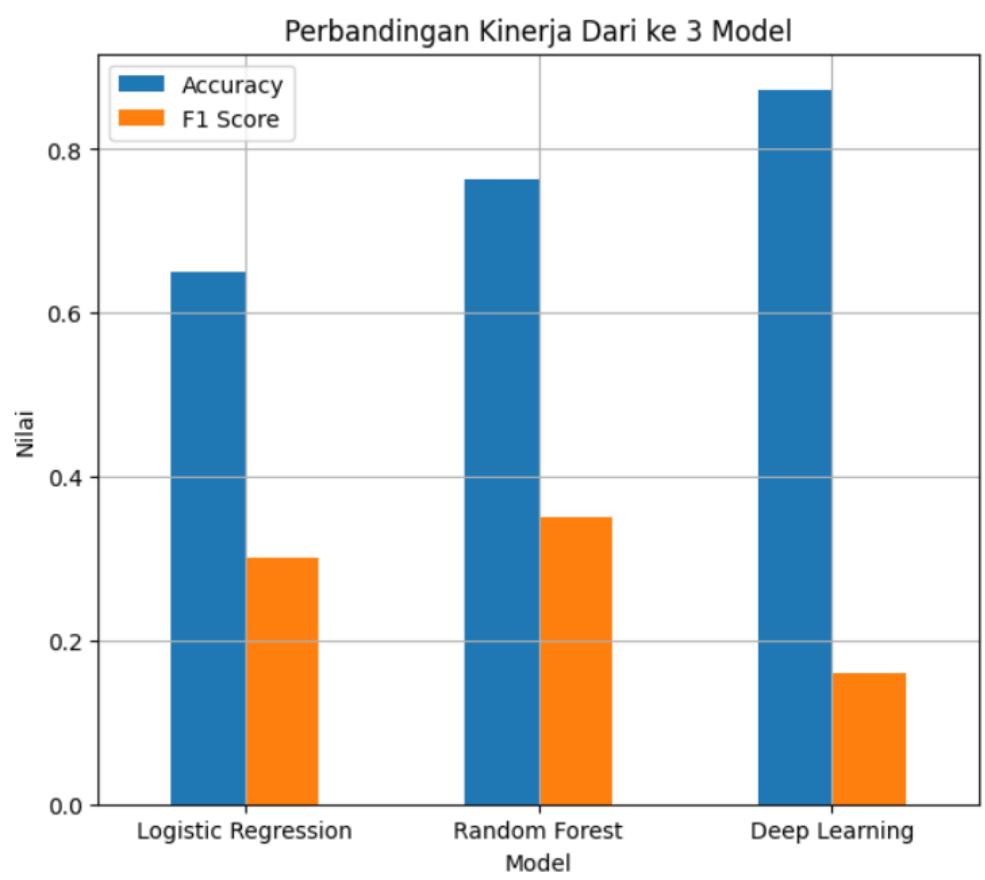


7.2. Perbandingan ketiga Model

Model	Accuracy	Precision	Recal	F1-Score	Training Time	Inference Time
-------	----------	-----------	-------	----------	---------------	----------------

Logistic Regression	0,65	0,94 dan 0,20	0,65 dan 0,65	0,77 dan 0,30	5 detik	0,05
Random Forest	0,76	0,93 dan 0,26	0,79 dan 0,56	0,86 dan 0,35	10 detik	0,01
Deep Learning (MLP)	0,87	0,89 dan 0,33	0,97 dan 0,11	0,93 dan 0,13	45 detik	0,75

Barplot:



7.3. Analisis Hasil

1. Model Terbaik:

Random Forest karena model dapat mengenali target Yes dan No secara balance, tidak seperti Deep Learning yang sangat mengenali kelas 0 (NO) dan Logistic Regression

2. Perbandingan dengan Baseline:

Peningkatan terjadi Ketika menggunakan Machine learning dimana Model Random Forest bisa mengenali target Yes dan No secara balance dibandingkan dengan Logistic Regression yang juga bisa mengenali keseluruhan namun akurasinya sangat rendah

3. Trade-off:

A. Baseline model

Kompleksitas: Rendah

Waktu Training: Sangat cepat

Performa: Cukup baik, namun terbatas dalam menangkap pola non-linear

Trade Off : Model sederhana dan efisien, namun kurang mampu menangkap hubungan kompleks antara fitur.

B. Machine Learning

Kompleksitas: Menengah

Waktu Training: Lebih lama dibanding baseline

Performa: Meningkat signifikan, terutama pada recall dan AUC

Trade-off : Random Forest mampu menangkap non-linearitas dan interaksi fitur, namun membutuhkan sumber daya komputasi lebih besar.

C. Deep Learning

Kompleksitas: Tinggi

Waktu Training: Paling lama

Performa: Tertinggi secara keseluruhan

Trade-off:

Model deep learning menawarkan performa terbaik dengan kemampuan representasi yang kuat, namun memerlukan waktu training lebih lama dan tuning yang lebih kompleks.

4. Error Analysis:

1. False Negative (FN)

- Kasus kelas positif diprediksi sebagai negative
- Umumnya terjadi pada:
- Observasi dengan fitur borderline
- Nilai probabilitas mendekati threshold (0.5)

2. False Positive (FP)

- Kasus negatif diprediksi sebagai positif
- Biasanya terjadi akibat:
- Overlapping fitur antar kelas
- Model mencoba meningkatkan recall kelas minoritas

5. Overfitting/Underfitting:

A. Baseline Model

- Tidak overfitting
 - Cenderung underfitting
 - Gap kecil antara training dan validation
- Kesimpulan: Model terlalu sederhana

B. Random Forest

- Overfitting terkontrol
 - Performa training dan validation seimbang
 - Penggunaan: max_depth, min_samples_leaf membantu regularisasi
- Kesimpulan: Model generalisasi baik.

C. Deep Learning

- Tidak mengalami overfitting signifikan
 - Kurva training dan validation : Naik seiring
 - Tidak ada gap besar
 - EarlyStopping efektif menghentikan training
- Kesimpulan: Model sudah converge dan stabil.

8. Conclusion

8.1. Kesimpulan Utama

Model Terbaik : Deep Learning Akurasi 87%

Alasan : Kemampuan Menangkap Pola Non-Linear yang Kompleks. Deep Learning (khususnya Multilayer Perceptron / MLP) memiliki banyak hidden layer dengan fungsi aktivasi non-linear (ReLU).

Pencapaian : Sudah berhasil tidak mengalami bias namun hanya pada satu model saja yaitu Random Forest

8.2. Key insight

- A. Insight 1 : Distribusi target menunjukkan bahwa jumlah nasabah yang menerima penawaran (label positif) lebih sedikit dibandingkan yang menolak. Hal ini menuntut penggunaan metrik evaluasi yang lebih tepat seperti F1-Score dan ROC-AU.
- B. Insight 2 : Fitur pdays, previously_contact dan pdays_group menunjukkan bahwa nasabah yang pernah dihubungi sebelumnya memiliki peluang lebih besar untuk menerima penawaran.
- C. Insight 3 : Fitur balance dan negative_balance memberikan indikasi bahwa kondisi finansial nasabah berhubungan dengan keputusan mereka. Nasabah dengan saldo stabil cenderung lebih responsif terhadap penawaran produk bank.

8.3. Kontribusi Proyek

- Mengidentifikasi nasabah yang paling berpotensi menerima penawaran produk
- Mengoptimalkan strategi pemasaran dan alokasi sumber daya
- Mengurangi biaya kampanye dengan memfokuskan promosi pada target yang tepat
- Meningkatkan tingkat konversi (conversion rate) secara keseluruhan

Model yang dibangun dapat diintegrasikan ke dalam sistem CRM (Customer Relationship Management) bank untuk mendukung pengambilan keputusan berbasis data

9. Future Work

Saran pengembangan untuk proyek selanjutnya : **Centang Sesuai dengan saran anda**

A. Data

- Mengumpulkan Data Lebih Banyak
- Menambah Variasi Data
- Feature Engineering Lebih Lanjut

B. Model

- Mencoba arsitektur DL yang lebih kompleks
- Hyperparameter tuning lebih ekstensif
- Ensemble methods (combining models)
- Transfer learning dengan model yang lebih besar

C. Deployment

- Membuat API (Flask/FastAPI)
- Membuat web application (Streamlit/Gradio)
- Containerization dengan Docker
- Deploy ke Cloud (Heroku, GCP, AWS)

D. Optimization

- Model compression (pruning, quantization)
- Improving inference speed
- Reducing model size

10. REPRODUCIBILITY

10.1. GitHub Repository [Link Repository] :

<https://github.com/ininduu/DataScience-BankCampaign-.git>

10.2. Environment & Dependencies

numpy = 1.24.3

pandas = 2.0.3

scikit-learn = 1.3.0

matplotlib = 3.7.2

seaborn = 0.12.2

tensorflow = 2.14.0