**LAPORAN PROYEK PENGOLAHAN DATA BESAR**

***Classification in Bank Marketing for Deposit Subscription Prediction Using Spark Mlib with Logistic Regression Model***



**Disusun oleh:**

1. 12S17007 – Ernike Nelsi Manurung
2. 12S17012 – Reza Oktovian Siregar
3. 12S17020 – Jovan Pioma Pakpahan
4. 12S17021 – Inggrit S. Purba

**PROGRAM STUDI SARJANA SISTEM INFORMASI**

**FAKULTAS INFORMATIKA DAN TEKNIK ELEKTRO**

**INSTITUT TEKNOLOGI DEL**

**MEI 2021**

# DAFTAR ISI

[DAFTAR ISI i](#_Toc72697954)

[DAFTAR GAMBAR ii](#_Toc72697955)

[DAFTAR TABEL iii](#_Toc72697956)

[DAFTAR KODE PROGRAM iv](#_Toc72697957)

[BAB 1 PENDAHULUAN 1](#_Toc72697958)

[1.1 Latar Belakang 1](#_Toc72697959)

[1.2 Tujuan 2](#_Toc72697960)

[1.3 Manfaat 2](#_Toc72697961)

[1.4 Ruang Lingkup 3](#_Toc72697962)

[BAB 2 SOLUSI 6](#_Toc72697963)

[2.1 *Big Data Architecture* 6](#_Toc72697964)

[2.2 *Machine Learning Pipeline* 7](#_Toc72697965)

[2.2.1 ETL *Process* 8](#_Toc72697966)

[2.2.2 *Data Preparation* 8](#_Toc72697967)

[2.2.3 *Data Segregation* 8](#_Toc72697968)

[2.2.4 *Model Training* 9](#_Toc72697969)

[2.2.5 *Evaluation* 9](#_Toc72697970)

[2.2.6 *Tuning Model* 9](#_Toc72697971)

[BAB 3 IMPLEMENTASI 10](#_Toc72697972)

[3.1 *ETL* 10](#_Toc72697973)

[3.2 *Data Preparation* dan Exploratory Data Analysis (EDA) 10](#_Toc72697974)

[3.3 *Load Data* 20](#_Toc72697975)

[3.4 *Model Training* 21](#_Toc72697976)

[3.5 *Evaluation* 23](#_Toc72697977)

[3.6 *Tuning Model* 24](#_Toc72697978)

[BAB 4 HASIL DAN PEMBAHASAN 26](#_Toc72697979)

[4.1 Hasil Prediksi 26](#_Toc72697980)

[4.2 Hasil *Tuning Model* 27](#_Toc72697981)

[4.3 Hasil Visualisasi 28](#_Toc72697982)

[4.4 Kesimpulan 33](#_Toc72697983)

[DAFTAR PUSTAKA 34](#_Toc72697984)

# DAFTAR GAMBAR

[Gambar 1. Big Data Architecture 6](#_Toc72697988)

[Gambar 2. Machine Learning Pipeline 7](#_Toc72697989)

[Gambar 3. Boxplot outlier 17](#_Toc72697990)

[Gambar 4. Hasil Prediksi 26](#_Toc72697991)

[Gambar 5. Jumlah Hasil Prediksi 26](#_Toc72697992)

[Gambar 6. Tingkat Akurasi 27](#_Toc72697993)

[Gambar 7. Hasil Prediksi dari Tuning Model 27](#_Toc72697994)

[Gambar 8. Hasil Evaluasi Tuning Model 27](#_Toc72697995)

[Gambar 9. Confusion Matrix without Normalization Tuning Model 28](#_Toc72697996)

[Gambar 10. Confusion Matrix with Normalization Tuning Model 28](#_Toc72697997)

[Gambar 11. Bank Marketing Predictions Dashboard 28](#_Toc72697998)

[Gambar 12. Visualisasi Poutcome and Campaign Based on Contact 29](#_Toc72697999)

[Gambar 13. Call Duration per Day of Week 30](#_Toc72698000)

[Gambar 14. Cons Conf Idx, Cons Price Idx, Nr Employed Based on Contact 31](#_Toc72698001)

[Gambar 15. Euribor3M per Day of Week 31](#_Toc72698002)

[Gambar 16. Count of Education for Each Type 32](#_Toc72698003)

[Gambar 17. Count of Loan Based on Marital/Housing 33](#_Toc72698004)

# DAFTAR TABEL

[Tabel 1. Keterangan Dataset 3](#_Toc72698008)

# DAFTAR KODE PROGRAM

[Kode Program 1. Supressing Warning 10](#_Toc72698024)

[Kode Program 2. Library Data Preprocessing 11](#_Toc72698025)

[Kode Program 3. Import and Read Dataset 11](#_Toc72698026)

[Kode Program 4. Menghitung Null Values 11](#_Toc72698027)

[Kode Program 5. Info Data 12](#_Toc72698028)

[Kode Program 6. Data Unik 13](#_Toc72698029)

[Kode Program 7. Mencari Nilai Null 13](#_Toc72698030)

[Kode Program 8. Menangani Unknown 14](#_Toc72698031)

[Kode Program 9. Remove Unknown pada Housing 14](#_Toc72698032)

[Kode Program 10. Bobot Unknown 14](#_Toc72698033)

[Kode Program 11. Current Value Housing 14](#_Toc72698034)

[Kode Program 12. Impute values 14](#_Toc72698035)

[Kode Program 13. Current Values dari Housing 15](#_Toc72698036)

[Kode Program 14. Mengatasi Unknown pada Loan 15](#_Toc72698037)

[Kode Program 15. Defenisikan data dengan df2 15](#_Toc72698038)

[Kode Program 16. Mengubah Data Kategorikal 16](#_Toc72698039)

[Kode Program 17. Proses One-Hot Encoding pada Data Kategorikal 16](#_Toc72698040)

[Kode Program 18. Drop Fitur yang Tidak Diperlukan 17](#_Toc72698041)

[Kode Program 19. Remove Outlier pada Age 18](#_Toc72698042)

[Kode Program 20. Remove Outlier pada Duration 18](#_Toc72698043)

[Kode Program 21. Remove Outlier pada Campaign 18](#_Toc72698044)

[Kode Program 22. Remove Outlier pada Previous 19](#_Toc72698045)

[Kode Program 23. Penggabungan data 19](#_Toc72698046)

[Kode Program 24. Final Data 19](#_Toc72698047)

[Kode Program 25. Load csv to Apache Spark 20](#_Toc72698048)

[Kode Program 26. Convert csv to Parquet File 20](#_Toc72698049)

[Kode Program 27. Load Parquet File From Hadoop into Apache Spark 21](#_Toc72698050)

[Kode Program 28. Persiapan Data untuk Machine Learning 21](#_Toc72698051)

[Kode Program 29. Transform Data 22](#_Toc72698052)

[Kode Program 30. Split Dataset 22](#_Toc72698053)

[Kode Program 31. Bangun Model Logistic Regression 22](#_Toc72698054)

[Kode Program 32. Membangun Arsitektur Pipeline 23](#_Toc72698055)

[Kode Program 33. Prediksi pada Test Data 23](#_Toc72698056)

[Kode Program 34. Menyimpan Hasil Prediksi dalam Bentuk .csv 23](#_Toc72698057)

[Kode Program 35. Melihat Jumlah Hasil Prediksi berdasarkan Label dan predictedLabel 23](#_Toc72698058)

[Kode Program 36. Menampilkan Nilai Akurasi 23](#_Toc72698059)

[Kode Program 37. Implementasi Create Param Grid 24](#_Toc72698060)

[Kode Program 38. Train Model with 5-fold CrossValidator 24](#_Toc72698061)

[Kode Program 39. Test Cross Validation Model 24](#_Toc72698062)

# PENDAHULUAN

## **Latar Belakang**

*Marketing Campaign* adalah susunan strategi aktivitas pemasaran yang dimanfaatkan untuk melakukan promosi produk melalui berbagai media seperti televisi, radio, telepon seluler maupun *platform online*. Didalam bidang *marketing* terdapat 2 pendekatan didalam periklanan dan promosi yakni *mass marketing* dan *direct marketing* [1]. Pada kasus didalam proyek ini akan dilakukan melalui media telepon seluler yang termasuk didalam *direct marketing.* Kegiatan *marketing* merupakan bagian penting didalam bisnis apapun, salah satunya adalah bank *marketing*. Kumpulan data *Bank Marketing* terkait dengan kampanye pemasaran langsung dari lembaga perbankan Portugis yang dimana kampanye tersebut didasarkan melalui panggilan telepon. Permasalahan yang sering terjadi pada kasus ini seperti ditemukannya lebih dari satu kontak ke klien yang sama diperlukan, untuk mengakses apakah produk (deposito berjangka/bank term deposit) akan ('ya') atau ('tidak') berlangganan. Didalam *Bank Direct Marketing* ditemukan aspek yang penting yakni bagaimana memperoleh akurasi yang tinggi didalam proses klasifikasi berdasarkan ketepatan informasi tertentu yang didapatkan dari pelanggan serta dianggap penting bagi pihak bank (Elsalamony, 2013).

Pihak Bank membutuhkan strategi untuk dapat mengelola data para nasabah yang berjumlah besar. Salah satu strategi yang dapat dilakukan untuk pengolahan informasi yang berasal dari data nasabah sebelumnya yakni dengan cara memprediksi. Melalui hasil prediksi tersebut dengan mengolah serta memanfaatkan data deposito nasabah yang sudah ada maka diharapkan akan menemukan informasi yang bersifat penting dan bernilai.

*Marketing* didalam *banking service* semakin kompleks diera Industri 4.0. Pada era ini sangat penting untuk menerapkan teknologi terbaru untuk dapat menyediakan pelayanan terhadap pengguna untuk menjadi lebih baik serta produktif. Semakin banyaknya data yang diperoleh memberikan tantangan baru terhadap penyimpanan data serta pemrosesan data. Salah satu *framework* yang dapat menangani masalah tersebut adalah Hadoop. Hadoop merupakan *platform* yang menyediakan penyimpanan data yang besar serta kemampuan komputasi secara terdistribusi untuk memproses data. Seiring berkembangnya teknologi, *framework* lainnya juga bermunculan, misalnya *Apache Spark, Hive. Apache Spark* merupakan perpaduan antara *machine learning* dan *big data* (*unified engine*) yang dapat melakukan pemrosesan untuk mengolah data yang besar dengan cepat. *Apache Spark* dapat berjalan diatas infrastruktur *Hadoop* untuk meningkatkan fungsionalitas. Selain itu, *Apache Spark* juga memiliki beberapa komponen yang mendukung proses pengklasifikasian, yaitu *Spark MLib* dimana *Spark MLib* menyediakan pustaka atau *library* untuk mengimplementasikan teknik *machine learning* seperti *classification, regression, clustering*, dll. Dalam proyek ini, akan menggunakan teknik *classification* untuk melakukan klasifikasi dimana algoritma klasifikasi yang akan digunakan adalah *Logistic Regression*. *Logistic Regression* merupakan suatu cara pemodelan masalah keterhubungan antara suatu variabel independen terhadap variabel dependen.

Oleh karena itu dalam proyek ini akan melakukan klasifikasi dengan algoritma *Linear Regression* untuk menentukan atribut data yang dapat dimanfaatkan dalam memprediksi apakah klien akan berlangganan (ya / tidak) deposito berjangka sebagai strategi didalam pemasaran dengan memanfaatkan teknologi dalam pemrosesan data yang besar.

## **Tujuan**

Adapun tujuan didalam melakukan penelitian pengolahan data besar ini yakni:

1. Memprediksi klien akan berlangganan deposito berjangka menggunakan metode klasifikasi pada kumpulan data *Bank Marketing* dengan menggunakan *Spark Mlib API.*

## **Manfaat**

Manfaat dari pengerjaan proyek ini yaitu:

1. Bagi Mahasiswa

Proyek ini diharapkan nantinya dapat memberikan pemahaman dan keterampilan bagi tim proyek untuk mengimplementasikan teknik *machine learning (scalable machine learning)* pada data besar khususnya pada salah satu metode dari pengklasifikasian yaitu *Logistic Regression* dengan teknologi pengolahan data besar menggunakan Spark Mlib API untuk memproses dan menganalisis data besar.

1. Pihak Bank

Proyek ini diharapkan dapat digunakan oleh pihak bank untuk mengelola data para nasabah yang berjumlah besar agar dapat memprediksi apakah klien akan berlangganan (ya / tidak) deposito berjangka sebagai strategi di dalam pemasaran. Melalui hasil prediksi tersebut dengan mengelola serta memanfaatkan data-data nasabah deposito yang sudah ada tersebut maka diharapkan akan menemukan informasi yang bersifat penting dan bernilai.

## **Ruang Lingkup**

Ruang lingkup dalam pengerjaan proyek ini adalah menggunakan salah satu metode dari *Classification* yaitu *Logistic Regression. Dataset* yang digunakan adalah data *Bank Marketin*g yang diunduh dari UC *Irvine Machine Learning Repository*. Dataset tersebut terdiri dari 21 atribut, sebagai berikut:

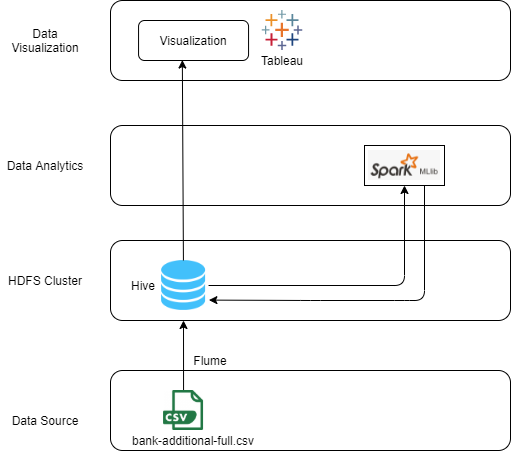
**Tabel 1. Keterangan Dataset**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Attribut** | **Tipe Atribut** | **Deskripsi** | **Values** |
| 1. | Age | Numerik | Usia dari setiap customer |  |
| 2. | Job | Kategorikal | Pekerjaan customer | 'admin.','blue-collar','entrepreneur','housemaid', 'management','retired','self-employed','services','student','technician','unemployed','unknown |
| 3. | Martial | Kategorikal | Kategori status perkawinan customer | 'divorced','married','single','unknown' |
| 4. | Education | Kategorikal | Jenis tingkatan pendidikan customer | 'basic.4y','basic.6y','basic.9y','high.school','illiterate','professional.course','university.degree','unknown' |
| 5. | Default | Kategorikal | Menunjukkan apakah ada kredit secara default? | 'no','yes','unknown' |
| 6. | Housing | Kategorikal | Menujukkan apakah mempunyai pinjaman perumahan? | 'no','yes','unknown' |
| 7. | Loan | Kategorikal | Menujukkan apakah mempunyai pinjaman pribadi? | 'no','yes','unknown' |
| 8. | Contact | Kategorikal | Jenis komunikasi kontak | 'cellular','telephone' |
| 9. | Month | Kategorikal | Terkait dengan kontak terakhir dengan kampanye pada bulan dalam setahun ini | 'jan', 'feb', 'mar', ..., 'nov', 'dec' |
| 10. | day\_of\_week | Kategorikal | Jenis hari terkait dengan kontak terakhir dalam seminggu ini | 'mon','tue','wed','thu','fri' |
| 11. | duration | Numerik | Durasi kontak terakhir, dalam detik (numerik). Atribut ini sangat mempengaruhi target keluaran (misalnya, jika durasi = 0 maka y = 'tidak'). Namun, durasinya tidak diketahui sebelum panggilan dilakukan. Setelah panggilan berakhir, y jelas diketahui. Dengan demikian, input ini hanya boleh dimasukkan untuk tujuan benchmark dan harus dibuang jika tujuannya adalah untuk memiliki model prediksi yang realistis. |  |
| 12. | Campaign | Numerik | Jumlah kontak yang dilakukan selama kampanye ini. |  |
| 13. | Pdays | Numerik | Jumlah hari yang berlalu setelah klien terakhir dihubungi dari kampanye sebelumnya (numerik; 999 berarti klien sebelumnya tidak dihubungi) |  |
| 14. | Previous | Numerik | Jumlah kontak yang dilakukan sebelum kampanye ini. |  |
| 15. | poutcome | Kategorikal | Hasil dari kampanye pemasaran sebelumnya | 'failure','nonexistent','success' |
| 16. | emp\_variation\_rate | Numerik | Tingkat variasi pekerjaan - indikator triwulanan (numerik). |  |
| 17. | Consumer\_price\_idx | Numerik | Indeks harga konsumen - indikator bulanan (numerik) |  |
| 18. | Consumer\_conf\_idx | Numerik | Indeks *confident* konsumen - indikator bulanan (numerik) |  |
| 19. | Euribor3m | Numerik | Euribor 3 bulan rate - indikator harian (numerik) |  |
| 20. | Nr\_employee | Numerik | Jumlah karyawan - indikator triwulanan (numerik) |  |
| 21. | y | Biner | Terkait dengan status customer apakah customer sudah berlangganan deposito berjangka? Ini bukan atribut, melainkan label observasi | ‘yes’, ‘no’ |

# SOLUSI

## ***Big Data Architecture***

Berikut ini merupakan arsitektur big data yang menjadi solusi terhadap permasalahan untuk memprediksi apakah klien akan berlangganan (ya/tidak) deposito jangka panjang.

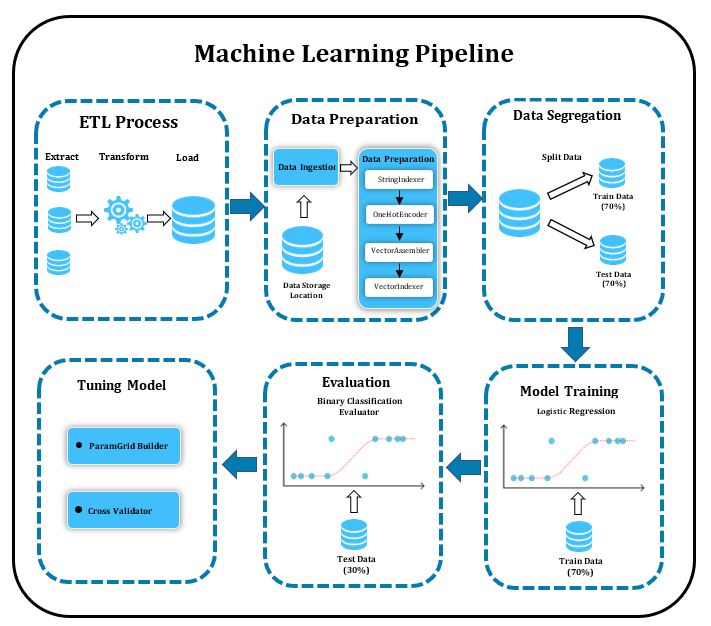


**Gambar 1. Big Data Architecture**

Dataset yang akan digunakan adalah dataset bank-additional-full dalam format CSV dimana CSV merupakan jenis data yang *semi-structured* sehingga untuk meload data menggunakan Apache Flume. Apache Flume merupakan service terdistribusi yang dirancang untuk menangkap, mengubah, dan meng-*ingest* data ke dalam HDFS. Dengan Apache Flume, *dataset* tersebut akan disimpan dalam Avro format, kemudian menggunakan Impala untuk mengkonversi Avro format ke Parquet format. Avro merupakan *row-based storage format* dimana semua data akan digunakan dalam pemrosesan sedangkan Parquet merupakan *column-based storage format* sehingga lebih sesuai digunakan pada *column-oriented query*, misalnya ketika hanya beberapa kolom yang akan dianalisis. Selain itu, pengeksekusian dengan Parquet 2 kali lebih cepat dibandingkan dengan Avro [2]. Oleh karena itu dilakukan pengkonversian dari Avro ke Parquet menggunakan Impala karena flume tidak menyediakan konfigurasi untuk mengubah ke Parquet format. Data yang telah dalam bentuk Parquet selanjutnya disimpan ke dalam Hive. Kemudian akan dilakukan *data analytics* menggunakan Spark MLib pada *dataset* yang telah diload ke Hive. Spark MLib menyediakan *library* *machine* *learning* yang dapat digunakan untuk melakukan prediksi, klasifikasi, clustering, dll. Dalam hal ini, *library machine learning* yang akan digunakan adalah Linear Regression untuk melakukan prediksi terhadap bank-additional-full dataset apakah klien akan berlangganan (ya / tidak) deposito berjangka. Hasil prediksi yang diperoleh dari *data analytics* yang telah dilakukan kemudian akan disimpan ke dalam Hive untuk selanjutnya divisualisasikan dengan Tableau. Visualisasi tersebut dapat memberikan pemahaman mengenai pola yang ditemukan dari hasil prediksi yang telah diperoleh dan memperoleh informasi dengan mudah.

## ***Machine Learning Pipeline***

Berikut ini merupakan *machine learning pipeline* yang menjadi solusi terhadap permasalahan untuk memprediksi apakah *customer* akan berlangganan deposito jangka panjang atau tidak.



**Gambar 2. Machine Learning Pipeline**

### **ETL *Process***

Proses ETL merupakan proses integrasi data dengan menggabungkan data dari berbagai sumber ke dalam suatu penyimpanan dan dimuat ke dalam *warehouse* atau sistem lainnya.

#### **Extract**

Hal pertama yang dilakukan adalah mengeskstraksi atau mengindektifikasi semua data dari sumbernya sebelum dipindahkan ke tempat penyimpanan yang lain.

#### **Transform**

Setelah tahap ekstraksi selesai, data perlu dipindahkan ke sistem target/ perantara untuk diproses lebih lanjut. *Transform* merupakan proses membersihkan dan mempersiapkan agregasi untuk analisis. Hal ini dilakukan untuk membantu memastikan data yang akan diolah siap dan kompatibel. Hal-hal yang dilakukan pada saat *transform* adalah menghapus data yang tidak konsisten dan data yang sama, menghapus data yang tidak dapat digunakan, mengatur data menurut jenisnya serta menambahkan aturan tertentu yang dapat meningkatkan kualitas data.

#### **Load**

*Load* merupakan tahap terakhir dari proses ETL. Load merupakan proses untuk memuat atau memindahkan data yang telah ditransformasi pada proses sebelumnya ke *data warehouse*. Data dapat dimuat sekaligus maupun secara terjadwal.

### ***Data Preparation***

*Data segregation* merupakan tahap dimana data dipersiapkan dan dianalisis dengan cara melakukan eksplorasi data sehingga dapat diketahui model atau algoritma yang cocok digunakan untuk mengolah data. Adapun hal yang dilakukan pada saat data preparation adalah EDA (Exploratory Data Analysis) yaitu dengan mengekstrak variabel atau fitur yang penting, menghapus variabel yang tidak diperlukan, memeriksa pencilan (*outliers*) dan nilai yang hilang (*missing values*).

### ***Data Segregation***

*Data segregation* adalah proses membagi (*split*) *dataset* menjadi dua bagian yaitu *train data* dan *test data. Train data* merupakan kumpulan data yang digunakan untuk melatih algoritma sementara *test data* merupakan kumpulan data yang digunakan untuk mengetahui performa algoritma yang sudah dilatih sebelumnya dengan model tertentu, sehingga dapat diketahui nilai keakuratannya. Pembagian dataset pada proyek ini menggunakan komposisi perbandingan 70% untuk *train data* dan 30% untuk *test data*. Antar *train data* dan *test data* tidak boleh overlap, karena jika overlap akan merusak proses training model. Sehingga komposisi untuk *train data* harus jauh lebih besar dari *test data*.

### ***Model Training***

Pada bagian ini dilakukan pembangunan model dengan suatu algoritma kemudian dilatih menggunakan *train data*. Algoritma yang digunakan adalah salah satu jenis klasifikasi yaitu *Logistic Regression*. *Logistic regression* adalah sebuah algoritma untuk mencari hubungan antara fitur dengan probabilitas hasil *output* diskrit tertentu. Algoritma ini digunakan untuk memprediksi hasil biner menggunakan *binomial logistic regression* atau memprediksi hasil *multiclass* dengan menggunakan *multinomial logistic regression*. Pada proyek ini akan menggunakan *binomial logistic regression*, hal ini dikarenakan *dataset* yang digunakan hanya memiliki 2 kelas yaitu pelanggan yang berlangganan deposito berjangka pendek dengan yang tidak (*yes/no*).

### ***Evaluation***

Setelah model dibangun dan dilatih menggunakan *train data*, selanjutnya dilakukan evaluasi untuk melihat performa atau nilai keakuratannya. Proses evaluasi dilakukan dengan memanfaatkan *test data*. *Output* dari proses ini adalah dapat melihat jumlah label ‘no’ diprediksi ‘no’, label ‘no’ diprediksi ‘yes’, label ‘yes’ diprediksi ‘yes’, serta label ‘yes’ diprediksi ‘no’, melihat hasil prediksi serta nilai keakuratannya.

### ***Tuning Model***

*Tuning model* merupakan proses untuk menentukan nilai parameter-parameter atau variabel dari model yang sudah dibuat sebelumnya. Pada proyek ini tuning model dilakukan dengan menggunakan ParamGridBuilder dan CrossValidator. ParamGridBuilder merupakan pendekatan *tuning hyperparameter* dengan membangun dan mengevaluasi model untuk setiap kombinasi parameter yang dispesifikasikan pada grid. Sementara CrossValidation merupakan pendekatan *tuning* *hyperparameter* dengan membagi data menjadi *data train* dan *data test* secara bergantian sesuai jumlah k yang dispesifikasikan.

# IMPLEMENTASI

## ***ETL***

Dalam proyek ini, proses ETL tidak dilakukan, karena menggunakan dataset yang sudah ada.

## ***Data Preparation* dan Exploratory Data Analysis (EDA)**

Pada tahapan awal pada pengembangan penelitian ini dilakukan terlebih dahulu proses persiapan data serta menganalisis data dengan cara melakukan eksplorasi data untuk nantinya peneliti dapat memutuskan bagaimana cara maupun metode yang digunakan untuk mengolah data. Pada proses *data preparation* sebagai proses yang dilakukan untuk mengolah data mentah menjadi data yang memiliki nilai. Setelah itu akan dilakukan EDA sebagai proses yang penting sebelum melakukan *featuring engineering* dan proses membangun model karena dengan melakukan EDA dapat membantu peneliti memahami kondisi dari *dataset* yang dimiliki. Proses awal ini dilakukan untuk dapat:

1. Mengekstrak variabel penting dan meninggalkan variabel yang tidak berguna
2. Mengidentifikasi pencilan (*outliers*) dan nilai yang hilang (*missing values*)
3. Memahami hubungan antar variabel

Pada bagian dibawah ini dilakukan sebagai langkah untuk mengabaikan peringatan penghentian didalam Python.

*# Supressing warnings*

**import** **warnings**

warnings.filterwarnings('ignore')

**Kode Program 1. Supressing Warning**

Langkah selanjutnya melakukan proses menggunakan (meng-*import*) pustaka-pustaka yang diperlukan untuk proses *data preparation* dan *exploratory data analysis*. Pada kasus ini *library* yang digunakan dalam tahapan awal seperti numpy (dapat menyimpan data dalam *array*, memudahkan mengelola data *numeric* serta digunakan untuk operasi matriks dan *vector*), pandas (menyediakan struktur data dan proses analisis data), matplotlib (mendukung visualisasi data pada Python), seaborn (menyediakan antar-muka tingkat tinggi berlandaskan pada matplotlib.), dsb.

*# importing all the libraries used in the case study*

**import** **numpy** **as** **np**

**import** **pandas** **as** **pd**

**import** **matplotlib.pyplot** **as** **plt**

**import** **seaborn** **as** **sns**

**Kode Program 2. Library Data Preprocessing**

Proses selanjutnya akan dilakukan *read data* yang dimana data akan dibaca dari *dataset* bank-additional-full.csv. Dikarenakan format *dataset* berupa file CSV (*Comma Separated Values*) sehingga memiliki struktur untuk mengatur datanya yakni berupa koma. Oleh karena itu, agar data dapat dibaca dan dikelola maka data akan dipisah sesuai tanda koma menggunakan *delimiters*. Untuk menampilkan semua kolom kerangka data bank maka digunakan algoritma pd.options.display.max\_columns=None. Selanjutnya data yang telah dipisahkan setiap attributnya/dikelola maka *dataset* bank-additional-full.csv didefenisikan dengan penamaan yakni data. Hasil dari defenisi data akan ditampilkan seluruh atribut yang berjumlah 21 atribut beserta dengan isi data dari setiap attribut.

*# importing data and reading it*

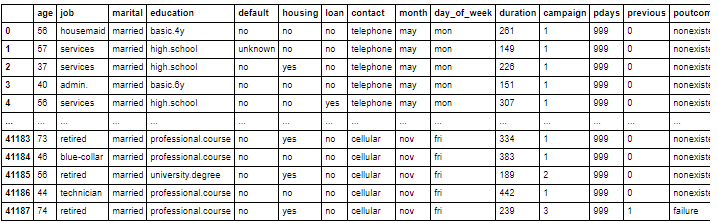
pd.options.display.max\_columns=**None**

data = pd.read\_csv('bank-additional-full.csv', delimiter=";")

data

**Kode Program 3. Import and Read Dataset**

*Output*:



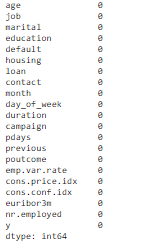
Pada bagian ini akan menghitung jumlah *null* yang terdapat pada 21 attribut pada data. Berdasarkan hasilnya terlihat bahwa jumlah yang bernilai null sebesar 0.

*# counting null values of every column*

data.isnull().sum()

**Kode Program 4. Menghitung Null Values**

*Output*:



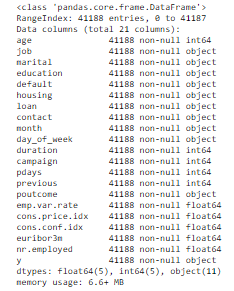
Pada bagian selanjutnya untuk menujukkan informasi data yang diteliti serta mencetak informasi tentang *DataFrame* termasuk jenis dan kolom indeks, nilai bukan *null*, dan penggunaan memori.

*# checking datatypes and null values in each columns*

data.info()

**Kode Program 5. Info Data**

*Output*:



Pada bagian ini akan menjelaskan terkait data-data yang unik yang terdapat pada setiap kolom attribut.

**for** col **in** data.select\_dtypes(include='object').columns:

print(col)

print(data[col].unique())

**Kode Program 6. Data Unik**

*Output*:



Pada bagian ini akan menjelaskan bahwa data yang dikelola apakah memiliki data yang null. Berdasarkan hasil *output* terlihat bahwa data tidak memiliki data yang *null*.

*# find missing values*

features\_na = [features **for** features **in** data.columns **if** data[features].isnull().sum() > 0]

**for** feature **in** features\_na:

print(feature, np.round(df[feature].isnull().mean(), 4), ' % missing values')

**else**:

print("No missing value found")

**Kode Program 7. Mencari Nilai Null**

*Output:*



Dikarenakan proses klasifikasi terdiri dari 2 pilihan ya dan tidak, maka peneliti akan mengelola data yang berisi dengan *unknown*. Maka peneliti akan mengelola data-data yang berisi *unknown* untuk dapat dimanfaatkan dalam perancangan model dengan cara mengambil data (variabel) dan mengembalikan yang dipilih secara acak jika data bernilai *unknown.*

*# write a function that takes a variable and returns a randomly selected # value if the variable is 'unknown'*

**def** impute\_values(variable, values, prob):

**if** variable == 'unknown':

**return** np.random.choice(values, p=prob)

**else**:

**return** variable

**Kode Program 8. Menangani Unknown**

Salah satu data yang mengandung data *unknown* yakni *housing.* Langkah awalnya akan mengambil data yang berasal dari data *housing* yang dimana data yang bernilai *unknown* akan di *remove.*

*# get a unique list values and drop 'unknown'*

values = list(set(data.housing.values))

values.remove('unknown') values

**Kode Program 9. Remove Unknown pada Housing**

Selanjutnya pada bagian ini sebagai proses penenetuan bobot pada setiap *value* (data) didalam kategori yang didefenisikan dengan penamaan prob.

*# determine the weight of each value in the category*

prob=data[data['housing']!='unknown'].housing.value\_counts(normalize=**True**) prob = [i/sum(prob) **for** i **in** prob]

prob

**Kode Program 10. Bobot Unknown**

Pada bagian ini akan dilakukan pengecekan terhadap distribusi pada *value,* sebagai berikut

*# check the current distribution of values*

data.housing.value\_counts(normalize=**True**)

**Kode Program 11. Current Value Housing**

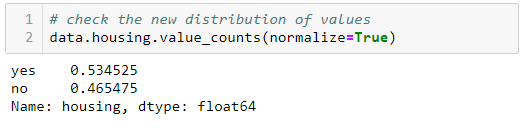
Maka pada langkah selanjutnya akan menghubungkan nilai untuk menangani nilai *unknown* dengan data yang dihitung dari data (fungsi *impute*). Salah satu pendekatan untuk memasukkan fitur kategorikal adalah dengan mengganti nilai yang hilang dengan kelas yang paling umum. Peneliti melakukannya dengan mengambil indeks dari fitur paling umum yang diberikan dalam fungsi value\_counts Pandas.

*# impute the values for 'unknown'*

data.housing = data.housing.map(**lambda** x: impute\_values(x, values, prob))

**Kode Program 12. Impute values**

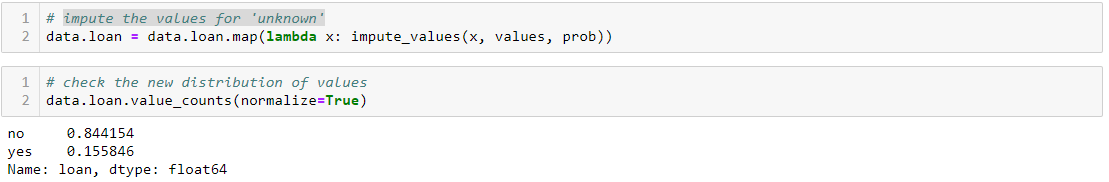
Lalu berikut hasil distribusi dari nilai yang baru:



**Kode Program 13. Current Values dari Housing**

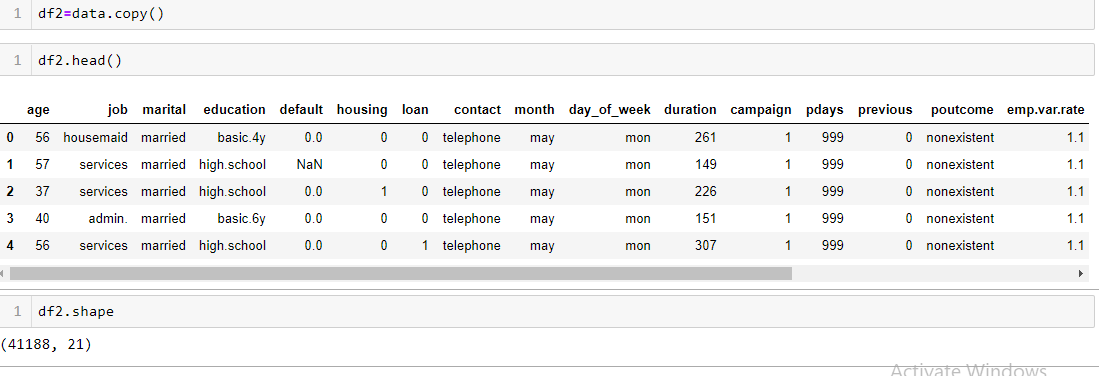
Mengatasi *unknown* yang terdapat pada *loan* juga dilakukan hal yang sama dengan cara mengatasi *unknown* pada *Housing.*





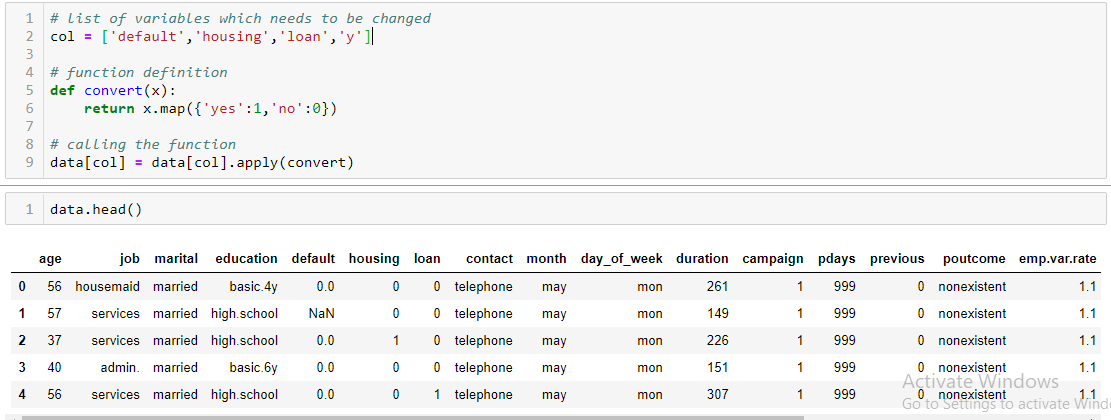
**Kode Program 14. Mengatasi Unknown pada Loan**

Selanjutnya tim peneliti mendefinisikan data yang sudah diolah dengan penamaan df2,



**Kode Program 15. Defenisikan data dengan df2**

Selanjutnya proses yang dilakukan adalah tranformasi/mengubah data kategorikal yakni default, housing, loan dan y menjadi data numerik yang dimana pilihan dari isi kolom direpresentasikan yes dengan 1 dan no dengan 0 untuk *no* sehingga data dapat dikelola oleh *machine learning.*



**Kode Program 16. Mengubah Data Kategorikal**

Setelah melakukan ekplorasi pada data numerik dan data kategorikal, tim peneliti akan mengelola data kategorikal dengan membuat variabel *dummy* untuk semua variabel kategorikal.

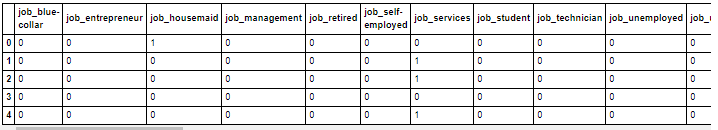
*# dummy variables of all categorical columns*

dummies = pd.get\_dummies(kolom\_kategorikal,drop\_first=**True**)

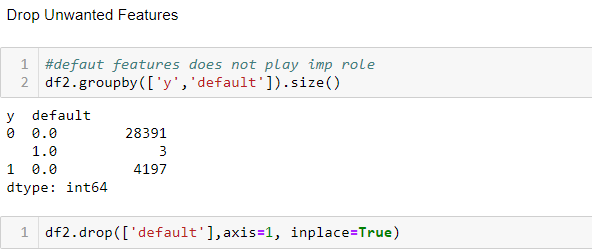
dummies.head()

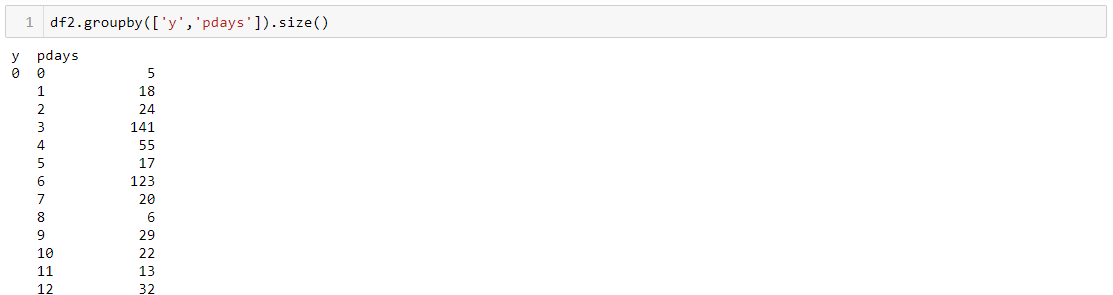
**Kode Program 17. Proses One-Hot Encoding pada Data Kategorikal**

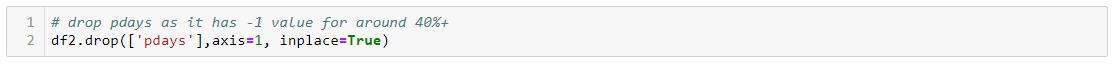
*Output:*



Proses selanjutnya melakukan *drop* pada fitur – fitur yang tidak diperlukan dalam membangun sistem prediksi yakni seperti default dan pdays.

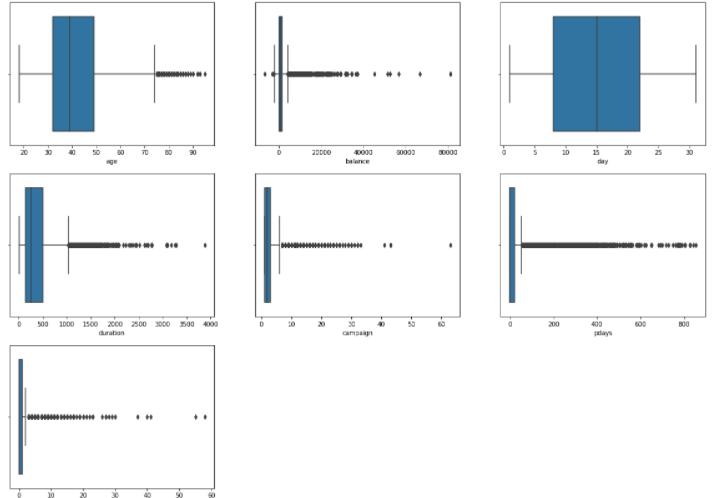






**Kode Program 18. Drop Fitur yang Tidak Diperlukan**

Selanjutnya tim peneliti akan melakukan *remove* terhadap *outliers* (data observasi yang muncul dengan nilai-nilai ekstrim/ mempunyai perilaku yang berbeda diabandingkan objek lain).

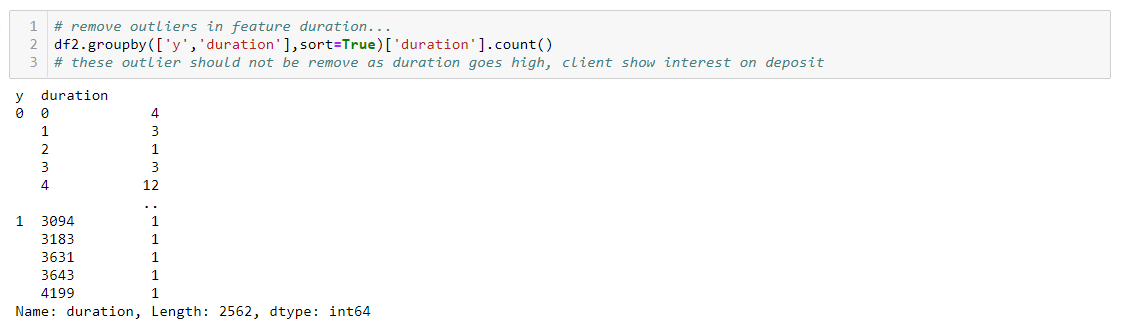


**Gambar 3. Boxplot outlier**

Berdasarkan boxplot terlihat bahwa *outlier* ditemukan pada kolom age*, balance, duration*, *compaign, pdays, previous.*



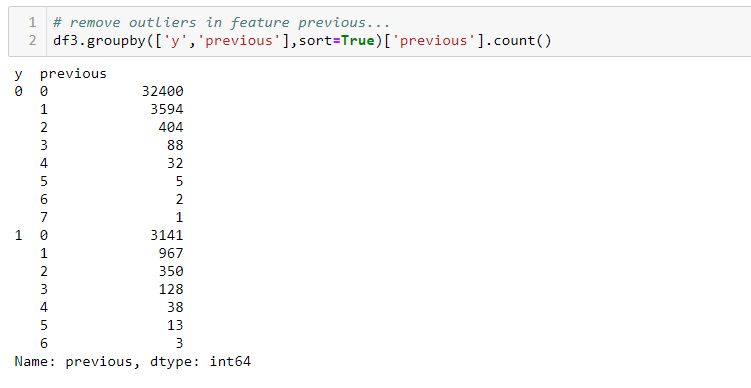
**Kode Program 19. Remove Outlier pada Age**



**Kode Program 20. Remove Outlier pada Duration**



**Kode Program 21. Remove Outlier pada Campaign**



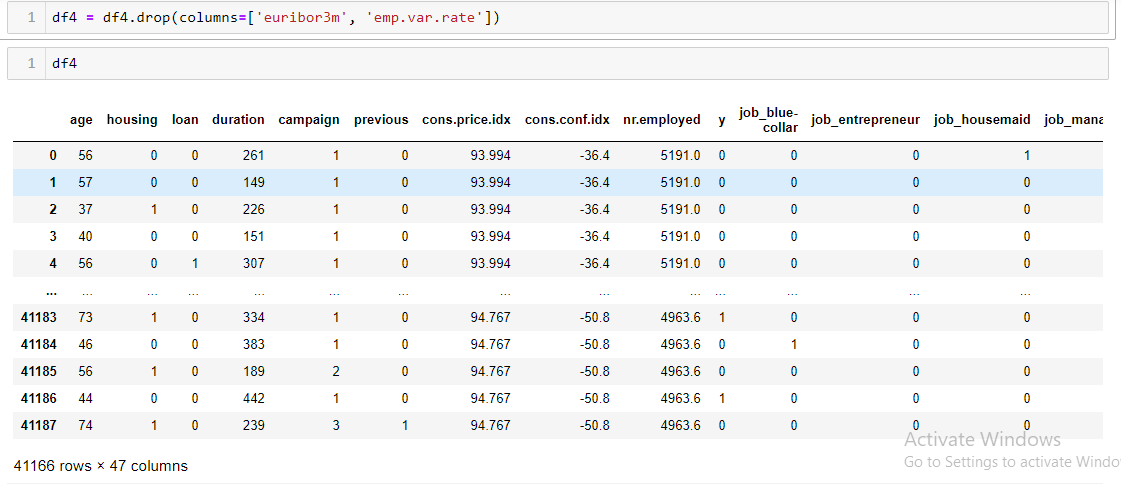
**Kode Program 22. Remove Outlier pada Previous**

Selanjutnya data digabungkan diantara data numerik beserta data kategorikal yang sudah dikelola pada proses sebelumnya sehingga nantinya seluruh isi kolom/attribute data berisi dengan numerik untuk dapat diproses *machine learning.*



**Kode Program 23. Penggabungan data**

Lalu tim peneliti kembali melakukan *drop* pada kolom yang tidak diperlukan yakni 'euribor3m'dan 'emp.var.rate; sehingga akan menghasilkan data sebagai berikut:



**Kode Program 24. Final Data**

## ***Load Data***

Implementasi *Load Data* pada *dataset* bank yang dilakukan mencakup pengubahan ekstensi *dataset* bank dari csv menjadi *parquet file.* Adapun proses ini dilakukan menggunakan *apache spark* melalui *jupyter notebook.* *File* csv di load melalui *apache spark*.

**from** **pyspark.sql.types** **import** StructType, StructField, StringType, IntegerType, FloatType

*# read csv from hdfs with specifying schema for all columns in data*

schema = StructType([

StructField("age", IntegerType(), **True**),

StructField("job", StringType(), **True**),

StructField("marital", StringType(), **True**),

StructField("education", StringType(), **True**),

StructField("housing", StringType(), **True**),

StructField("loan", StringType(), **True**),

StructField("contact", StringType(), **True**),

StructField("month", StringType(), **True**),

StructField("day\_of\_week", StringType(), **True**),

StructField("duration", IntegerType(), **True**),

StructField("campaign", IntegerType(), **True**),

StructField("previous", IntegerType(), **True**),

StructField("poutcome", StringType(), **True**),

StructField("emp.var.rate", FloatType(), **True**),

StructField("cons.price.idx", FloatType(), **True**),

StructField("cons.conf.idx", FloatType(), **True**),

StructField("euribor3m", FloatType(), **True**),

StructField("nr.employed", FloatType(), **True**),

StructField("y", StringType(), **True**),

])

*# load dataset from hdfs*

*#bank\_df = spark.read.csv("hdfs://localhost:9820/proyek/bank-additional-clean.csv", header=True, schema=schema);*

bank\_df = spark.read.csv("bank-additional-clean.csv", header=**True**, schema=schema);

**Kode Program 25. Load csv to Apache Spark**

Setelah file csv di load ke dalam *apache spark* kemudian data csv di baca seperti menampilkan isi data, menampilkan schema, melakukan *rename* pada kolom dan setelah itu *file* csv diubah menjadi *parquet file.*

*# change csv file to parquet hadoop format file and save it in hdfs*

*#bank\_df.write.parquet('hdfs://localhost:9820/proyek/bank-additional.parq')*

bank\_df.write.parquet('bank-additional.parq')

**Kode Program 26. Convert csv to Parquet File**

Setelah *file* csv diubah menjadi *parquet file* kode program tesebut juga akan melakukan *export* berupa *parquet file* yang akan di load ke dalam *hadoop system*. *Parquet file* yang di *export* di input ke dalam *hadoop system* dan dari *hadoop system* akan di masukkan menggunakan *apache spark*.

*# load parquet hadoop file format from hdfs*

*#df = spark.read.parquet("hdfs://localhost:9820/proyek/bank-additional.parq")*

df = spark.read.parquet("bank-additional.parq")

df.show(5)

df.printSchema()

**Kode Program 27. Load Parquet File From Hadoop into Apache Spark**

Pada Kode Program 27, *parquet file* yang sebelum nya di *export* ke *local* dan di input kedalam *hadoop system* kemudian dibaca menggunakan *apache spark*. Hal ini menandakan bahwa *parquet file* sudah dapat digunakan untuk diolah lebih lanjut.

## ***Model Training***

Selanjutnya adalah pemodelan dengan *logistic regression* menggunakan *train data*. Hal pertama yang dilakukan adalah mempersiapkan data untuk *Machine Learning*.

**def** get\_dummy(df, categoricalCols, continuousCols, labelCol):

indexers = [StringIndexer(inputCol=c, outputCol="**{0}**\_indexed".format(c)) **for** c **in** categoricalCols]

encoders = [OneHotEncoder(inputCol=indexer.getOutputCol(), outputCol="**{0}**\_encoded".format(indexer.getOutputCol())) **for** indexer **in** indexers]

assembler = VectorAssembler(inputCols=[encoder.getOutputCol() **for** encoder **in** encoders] + continuousCols, outputCol="features")

indexer = StringIndexer(inputCol=labelCol, outputCol='indexedLabel')

pipeline = Pipeline(stages = indexers + encoders + [assembler] + [indexer])

model=pipeline.fit(df)

data = model.transform(df)

data = data.withColumn('label', col(labelCol))

**return** data.select('age', 'job', 'marital', 'education', 'housing', 'loan', 'contact', 'month', 'day\_of\_week', 'duration', 'campaign', 'previous', 'poutcome', 'emp\_var\_rate', 'cons\_price\_idx', 'cons\_conf\_idx', 'euribor3m', 'nr\_employed', 'features', 'indexedLabel', 'label'), StringIndexer(inputCol='label').fit(data)

**Kode Program 28. Persiapan Data untuk Machine Learning**

Pada Kode Program 28 terdapat fungsi get\_dummy. Fungsi tersebut berfungsi untuk mengindeks setiap kolom yang bertipe data categorical dengan StringIndexer dan mengubahnya ke bentuk variabel yang diencode. Fungsi tersebut akan menghasilkan output yang memiliki vektor biner yang ditambahkan ke akhir setiap barisnya. Selanjutnya StringIndexer digunakan kembali untuk meng-encode semua label ke indeks label. Kemudian fungsi VectorAssembler digunakan untuk menggabungkan semua kolom fitur ke dalam kolom vektor. Index labels, menambahkan metadata ke kolom label dengan menggunakan StringIndexer lagi untuk meng-*encode* label ke indeks label. Selanjutnya terdapat pipeline yang digunakan untuk menyatukan beberapa transformer dan estimator.

*# transform the data*

categoricalColumns = ['job', 'marital', 'education', 'housing', 'loan', 'contact', 'month', 'day\_of\_week', 'poutcome']

numericCols = ['age', 'duration', 'campaign', 'previous', 'emp\_var\_rate', 'cons\_price\_idx', 'cons\_conf\_idx', 'euribor3m', 'nr\_employed']

(bank\_df, labelindexer) = get\_dummy(bank\_df, categoricalColumns, numericCols, 'y')

bank\_df.show(5)

**Kode Program 29. Transform Data**

Selanjutnya pada Kode Program 29 berfungsi untuk mengubah data yang sebelumnya ditentukan terlebih dahulu variabel yang bertipe data kategorikal dan numerikal serta kelasnya ‘y’.

*# split dataset to trainingData and testData*

(trainingData, testData) = bank\_df.randomSplit([0.7, 0.3], seed=10)

print("Training Dataset Count: " + str(trainingData.count()))

print("Test Dataset Count: " + str(testData.count()))

**Kode Program 30. Split Dataset**

Selanjutnya pada kode program diatas dataset di-*split* secara acak menjadi *train data* dan *test data*. Pembagian dataset di-*split* sebesar 70% untuk *train data* dan 30% untuk *test data*. Selanjutnya nilai *seed* sebesar 10, hal ini diset bertujuan untuk membuat hasil klasifikasi tetap sama jika dieksekusi berulang.

*# build logistic regression model*

lr = LogisticRegression(labelCol="indexedLabel", featuresCol="features")

**Kode Program 31. Bangun Model Logistic Regression**

Selanjutnya pada Kode Program 31, model *logistic regression* dibangun.

*#build Pipeline architecture*

*#convert indexed labels back to original labels*

labelConverter = IndexToString(inputCol="prediction", outputCol="predictedLabel", labels=labelindexer.labels)

*#chain indexers and tree in Pipeline*

pipeline = Pipeline(stages=[featureIndexer, lr, labelConverter])

*#train model*

lrModel = pipeline.fit(trainingData)

**Kode Program 32. Membangun Arsitektur Pipeline**

Kemudian pada Kode Program 32, arsitektur *pipeline* dibangun dengan mengubah kembali label yang diindeks ke label yang asli dan melatih model dengan menggunakan train data.

## ***Evaluation***

Setelah model selesai dilatih dengan menggunakan *train data*, maka selanjutnya adalah melihat performa atau nilai keakuratannya dengan memanfaatkan test data.  *# Make predictions on the test data using the transform() method.* predictions = lrModel.transform(testData) predictions.show(5)

**Kode Program 33. Prediksi pada Test Data**

Pada Kode Program 33, terdapat method transform() untuk membuat prediksi pada test data.

*# save bank\_addtional\_prediction to csv* **import** **pandas** **as** **pd** bank\_additional\_prediction = predictions.select('age', 'job', 'marital', 'education', 'housing', 'loan', 'contact', 'month', 'day\_of\_week', 'duration', 'campaign', 'previous', 'poutcome', 'emp\_var\_rate', 'cons\_price\_idx', 'cons\_conf\_idx', 'euribor3m', 'nr\_employed', 'features', 'label', 'probability', 'predictedLabel') bank\_additional\_prediction.toPandas().to\_csv('bank\_additional\_prediction.csv', header = **True**)

**Kode Program 34. Menyimpan Hasil Prediksi dalam Bentuk .csv**

Setelah dilakukan prediksi pada test data, hasil prediksi dapat diekspor dalam bentuk .csv dengan memanfaatkan *library* pandas. Hal ini dapat dilakukan dengan menentukan kolom apa saja yang ingin ditampikan seperti pada Kode Program 10.

*#group predictions by label and predictedLabel*

predictions.groupBy('label', 'predictedLabel').count().show()

**Kode Program 35. Melihat Jumlah Hasil Prediksi berdasarkan Label dan predictedLabel**

Kode Program 35 berfungsi untuk melihat jumlah label no diprediksi no, label no diprediksi yes, label yes diprediksi yes, dan label yes diprediksi no.

print("The Accuracy for test set is **{}**".format(cm.filter(cm.label == cm.predictedLabel).count()/cm.count()))

**Kode Program 36. Menampilkan Nilai Akurasi**

Selanjutnya pada Kode Program 36, akurasi model dapat dihitung dengan menghitung jumlah ketika label diklasifikasikan dengan benar.

## ***Tuning Model***

Pada bagian ini dijelaskan implementasi *tuning model* menggunakan ParamGridBuilder dan CrossValidator. *Grid search* merupakan pendekatan tuning hyperparameter yang membangun dan mengevaluasi model untuk setiap kombinasi parameter yang dispesifikasikan pada grid. Sementara itu *Cross Validation* merupakan pendekatan *tuning* *hyperparameter* dengan membagi data menjadi data train dan data test secara bergantian sesuai jumlah k yang dispesifikasikan. Berikut adalah implementasi *tuning* *hyperparameter*.

paramGrid = (ParamGridBuilder()

.addGrid(lr.regParam, [0.01, 0.5, 2.0])

.addGrid(lr.elasticNetParam, [0.0, 0.5, 1.0])

.addGrid(lr.maxIter, [1, 5, 10])

.build())

evaluator = BinaryClassificationEvaluator(rawPredictionCol="prediction",labelCol="indexedLabel")

**Kode Program 37. Implementasi Create Param Grid**

Kode Program 37 digunakan untuk membuat ParamGrid untuk Cross Validation dimana terdapat 3 *values* untuk regParam, 3 *values* untuk maxIter, dan 3 *values* untuk elasticNetParam, sehingga grid tersebut akan memiliki 3 x 3 x 3 = 27 parameter untuk CrossValidator. Selain itu, juga dispesifikasikan evaluator menggunakan BinaryClassificationEvaluator.

pipeline = Pipeline(stages=[featureIndexer, lr, labelConverter])

cv = CrossValidator(estimator=pipeline, estimatorParamMaps=paramGrid, evaluator=evaluator, numFolds=5, parallelism=10, seed=100)

cvModel = cv.fit(trainingData)

**Kode Program 38. Train Model with 5-fold CrossValidator**

Pada Kode Program 38 akan dilakukan train model menggunakan 5-fold CrossValidator pada *Pipeline architecture* yang telah dibuat.

predictions = cvModel.transform(testData)

predictions.select("features", "label", "probability", "predictedLabel").show(5)

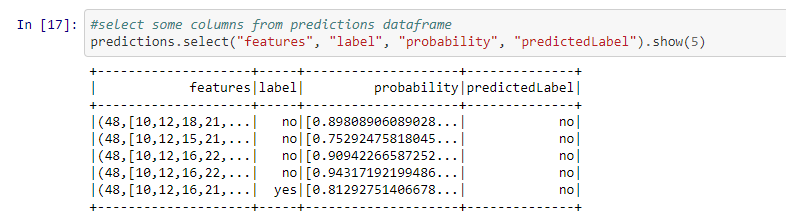
**Kode Program 39. Test Cross Validation Model**

Selanjutnya akan dilakukan prediksi mengunakan testData pada model yang telah di-train yaitu cvModel. cvModel tersebut merupakan model terbaik yang ditemukan dari CrossValidation saat train dilakukan. Prediksi dilakukan menggunakan method transform.

# HASIL DAN PEMBAHASAN

## **Hasil Prediksi**

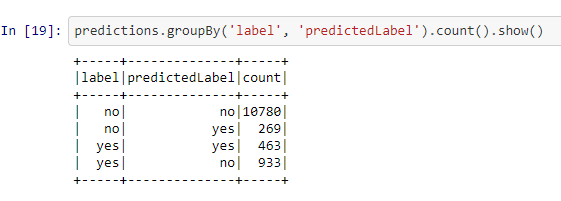
Hasil prediksi yang diperoleh dari setiap nasabah memiliki probabilitas yang berbeda seperti nasabah A memilih probabilitas yang berbeda dengan nasabah B dalam berlangganan pada bank tersebut.



**Gambar 4. Hasil Prediksi**

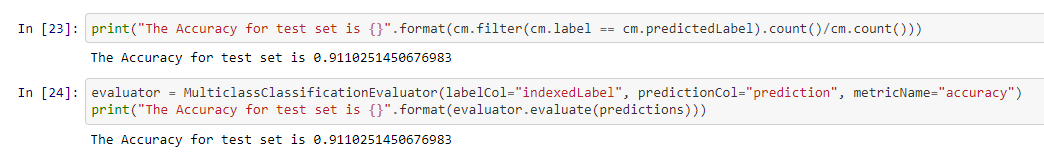
Gambar 4 merupakan hasil prediksi dari *dataset* bank yang telah diolah menggunakan *pipeline* dengan algoritma *Logistic Reggresion.* Dari hasil yang ditampilkan terdapat kolom *feature*, *label, probability, predicted label,* dimana *feature* akan menjelaskan karakteristik dari suatu nasabah baik dari *job, marital, education, housing, loan, contact, age, duration,* dll. Sedangkan *probability* merupakan probabilitas nasabah yang diprediksi akan berlangganan atau tidak pada bank tersebut dan dari probabilitas tersebut dihasilkan predicted label yang akan menjadi acuan terhadap bank untuk melakukan evaluasi terhadap marketing.

Pada Gambar 5 menunjukkan jumlah hasil prediksi. Dimana jumlah label ‘no’ diprediksi ‘no’ adalah 10780, jumlah label ‘no’ namun diprediksi ‘yes’ adalah 269, jumlah label ‘yes’ diprediksi ‘yes’ adalah 463 serta jumlah label ‘yes’namun diprediksi ‘no’ adalah 933.



**Gambar 5. Jumlah Hasil Prediksi**

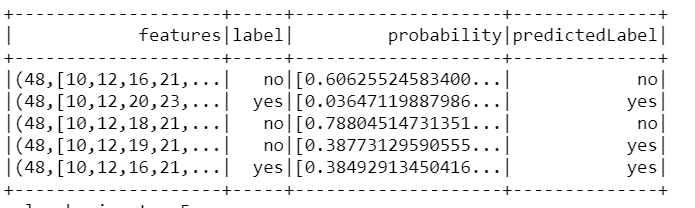
Sistem ini juga memiliki tingkat keakuratan yang memuaskan yaitu sekitar 91%.



**Gambar 6. Tingkat Akurasi**

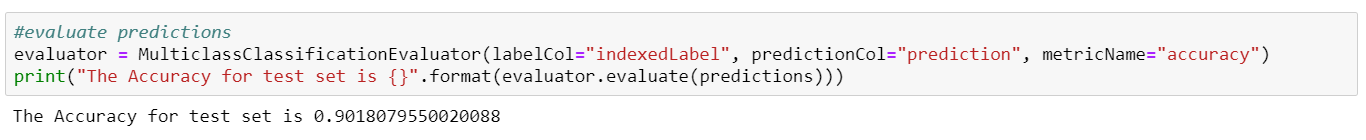
## **Hasil *Tuning Model***

Pada bagian ini dijelaskan mengenai hasil yang diperoleh dari tuning model dengan Param Grid dan Cross Validation yang telah dilakukan. Berikut adalah 5 sampel data hasil prediksi dari tuning model yang telah dilakukan.



**Gambar 7. Hasil Prediksi dari Tuning Model**

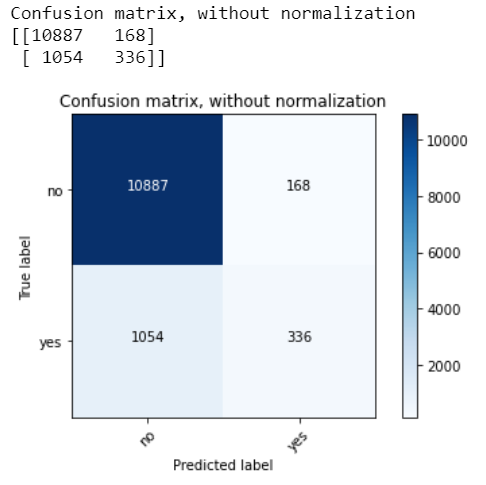
Dari hasil prediksi yang telah diperoleh tersebut, kemudian dilakukan evaluasi. *Accuracy* yang diperoleh dari model setelah dilakukan tuning adalah 0.90.



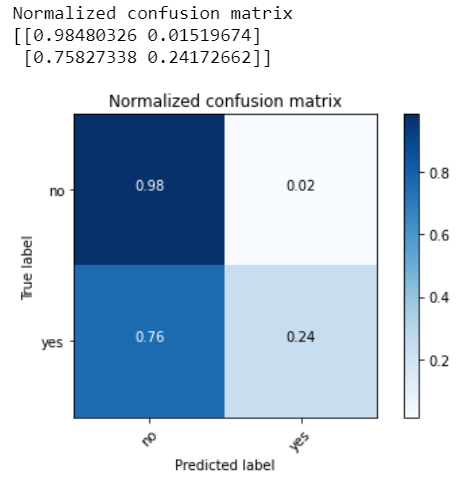
**Gambar 8. Hasil Evaluasi Tuning Model**

Jika dimodelkan dalam bentuk confusion matrix, hasil yang diperoleh ditunjukkan pada Gambar 7 dan Gambar 8 dimana terdapat:

* 1. Sebanyak 10887 (0.98) klien yang diprediksi tidak akan berlangganan deposit dan prediksi yang dihasilkan tepat.
  2. Sebanyak 168 (0.02) klien yang diprediksi akan berlanggan deposit namun sebenarnya tidak akan berlangganan deposit.
  3. Sebanyak 1054 (0.76) klien yang diprediksi tidak akan berlangganan deposit namun sebenarnya akan berlangganan deposit.
  4. Sebanyak 336 (0.24) klien yang diprediksi akan berlangganan deposit dan prediksi yang dihasilkan tepat.



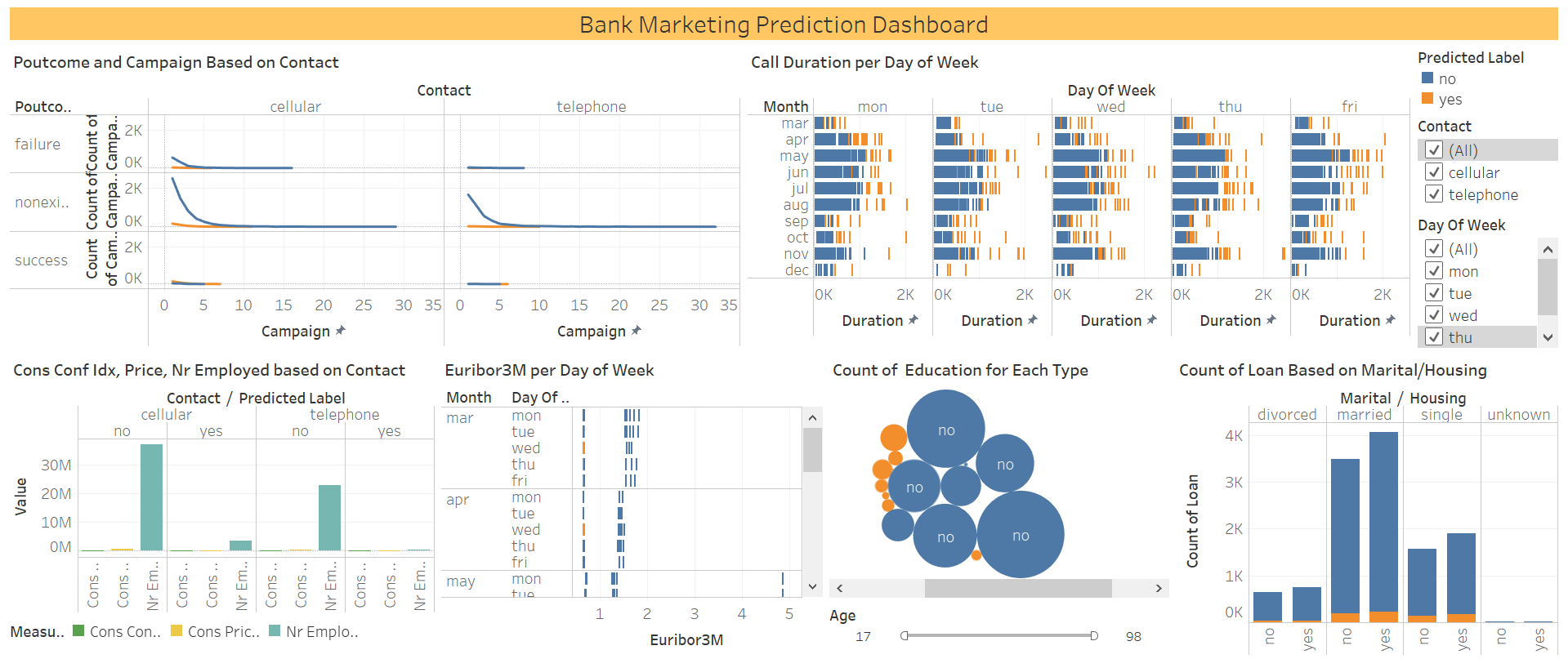
**Gambar 9. Confusion Matrix without Normalization Tuning Model**



**Gambar 10. Confusion Matrix with Normalization Tuning Model**

## **Hasil Visualisasi**

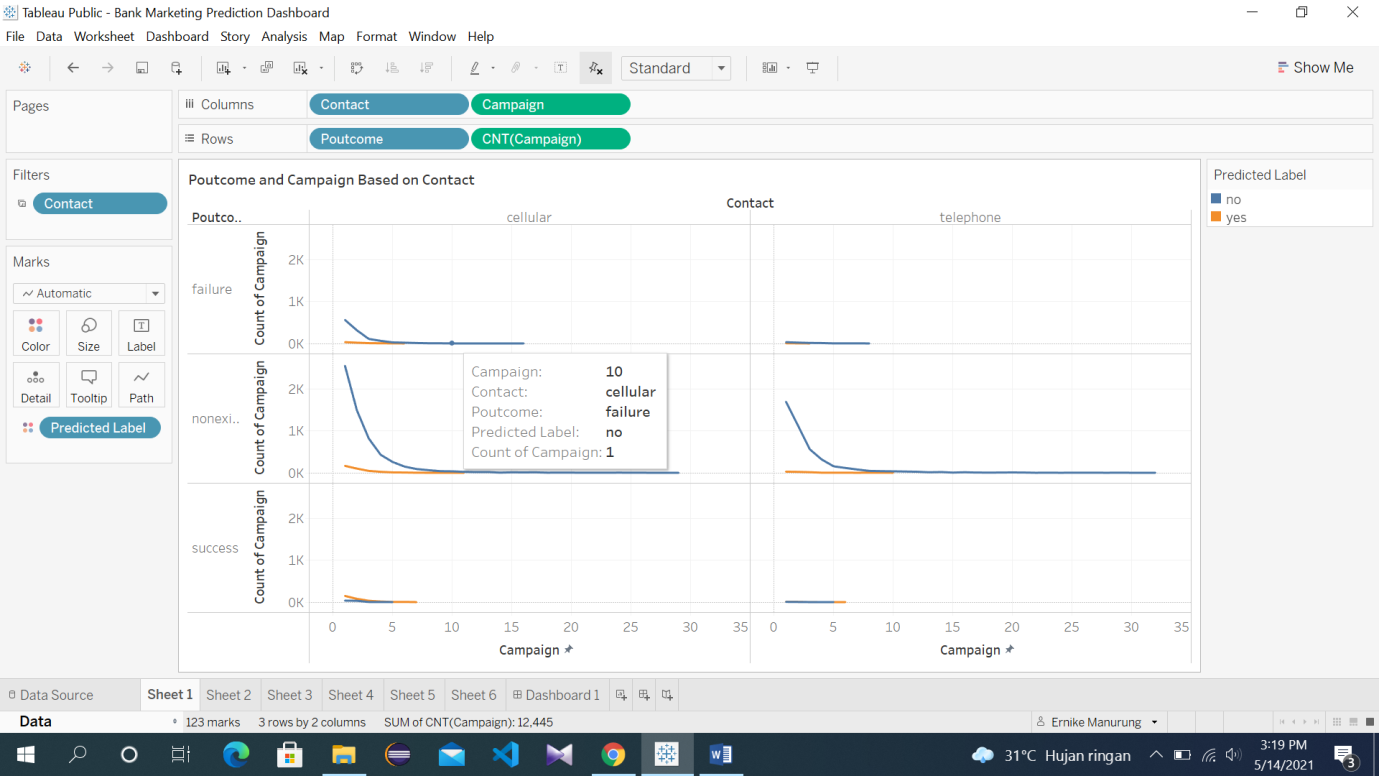
Hasil prediksi bank marketing analysis apakah klien akan berlanganan deposito atau tidak divisualisasikan dalam sebuah *dashboard* menggunakan *tool* Tableau yang ditunjukkan pada Gambar 10.



**Gambar 11. Bank Marketing Predictions Dashboard**

*Dashboard* tersebut terdiri dari 6 visualisasi. Berikut ini adalah penjelasan informasi yang dapat diperoleh dari masing-masing visualisasi tersebut.

1. Visualisasi *Poutcome* and *Campaign Based on Contact*



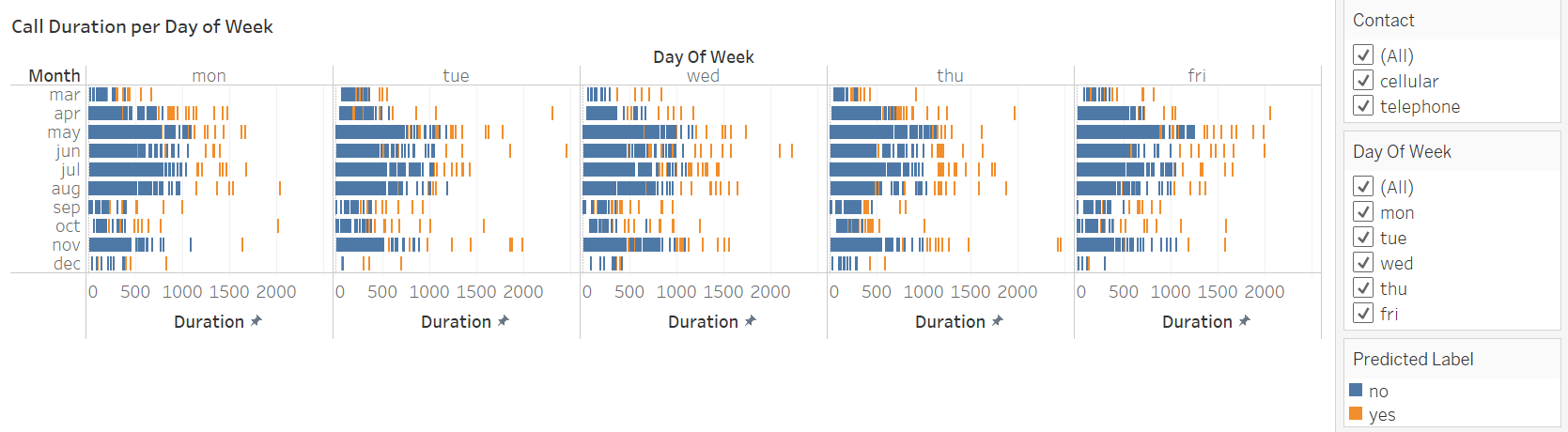
**Gambar 12. Visualisasi Poutcome and Campaign Based on Contact**

Visualisasi tersebut menampilkan hasil prediksi apakah klien berlangganan deposito atau tidak dari *poutcome*, *campaign*, yang dikelompokkan berdasarkan *contact*. Terdapat 3 nilai poutcome yang merupakan keluaran dari campaign (*failure, non-existent, success*). Pada sumbu x terdapat campaign yang merupakan banyaknya klien menelepon. Kemudian pada sumbu y terdapat *count of campaign* yang dikelompokkan berdasarkan nilai *poutcome*. Dapat dilihat bahwa:

1. *Contact cellular* dan *telephone* dengan *poutcome failure* memiliki jumlah yang lebih banyak dibandingkan *poutcome* yang *failure* dan *success*.
2. *Contact cellular* memiliki *count of campaign* yang lebih banyak pada setiap jenis *poutcome* dibandingkan *contact telephone.*
3. Untuk prediksi label apakah akan berlangganan deposito atau tidak dapat dilihat dari warna pada grafik, biru untuk “no” dan orange untuk “yes”. Pada *poutcome* *existent* dan *failure* memiliki grafik “no” yang lebih tinggi dan rata-rata *campaign* yang lebih banyak (>10) sementara *poutcome* success memiliki grafik “yes” lebih tinggi dibandingkan grafik “no” dengan rata-rata *campaign* (<10).

Berdasarkan hasil analisis dari visualisasi tersebut diperoleh bahwa *poutcome failure* dan *existent* memiliki peluang lebih besar untuk diprediksi tidak akan berlangganan deposito jangka panjang baik *contact celluler* maupun telepon dari ketinggian grafik ‘no’ dibandingkan ‘yes’. Sementara *poutcome success* dengan *contact celluler* diprediksi akan berlangganan deposito jangka panjang karena ketinggian grafik ‘yes’ dibandingkan ‘no’ pada *campaign* yang lebih kecil kemudian ketinggian grafiknya hampir sama sedangkan *contact telephone* pada *poutcome* *success* jumlah *campaign* untuk ‘yes’ atau ‘no’ relatif sama.

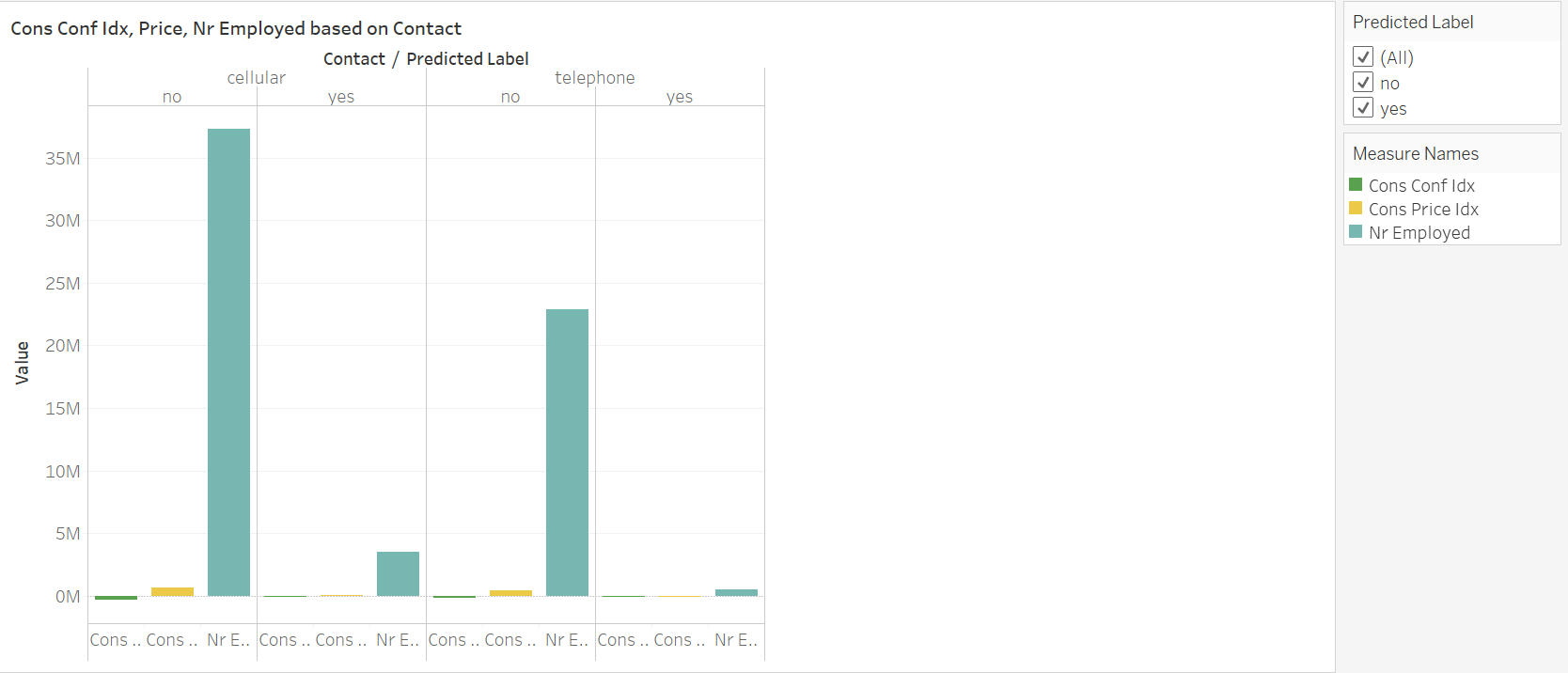
1. *Call Duration per Day of Week*



**Gambar 13. Call Duration per Day of Week**

Visualisasi tersebut menampilkan durasi panggilan *contact celluler* dan *telephone* perbulan dalam setiap hari. Diperoleh informasi bahwa:

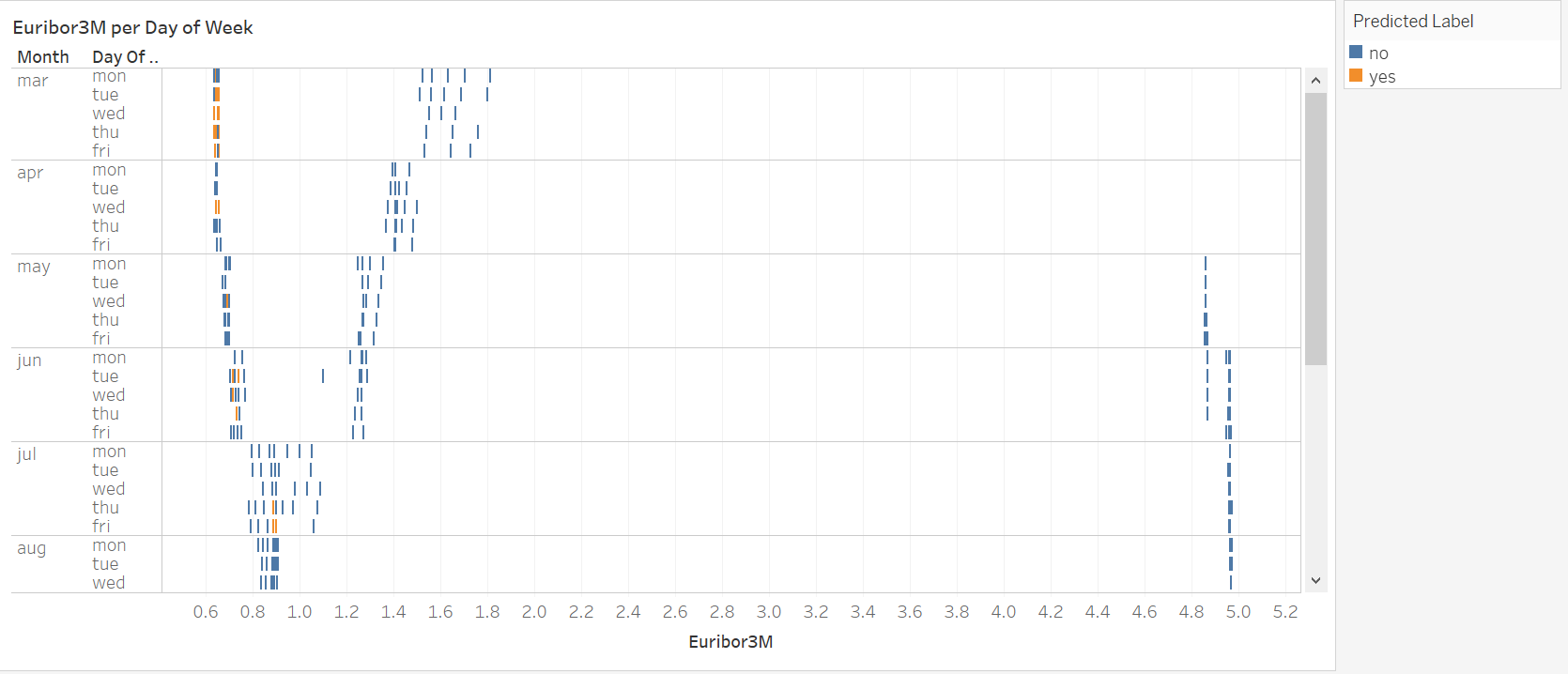
1. Durasi panggilan telepon paling banyak ada pada bulan Mei di setiap hari. Sementara yang paling jarang adalah di bulan Desember di setiap harinya.
2. Prediksi apakah klien akan berlangganan deposito atau tidak dapat dilihat berdasarkan penggunaan label warna biru untuk ‘no’ dan orange untuk ‘yes’.
3. Durasi panggilan telepon yang lebih sedikit cenderung diprediksi tidak akan berlangganan deposito sementara durasi panggilan telepon yang lebih lama diprediksi akan berlangganan deposito jangka panjang.
4. *Cons Conf Idx, Cons Price Idx, Nr Employed Based on Contact*

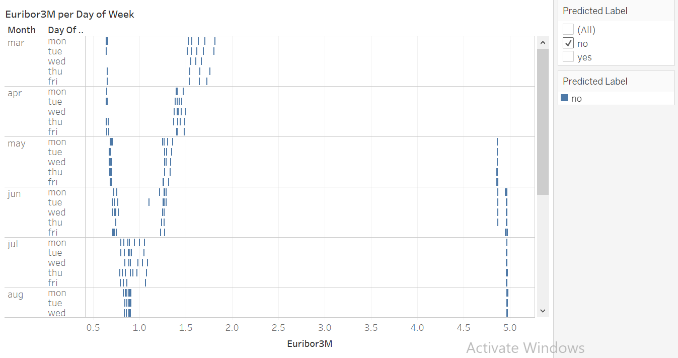
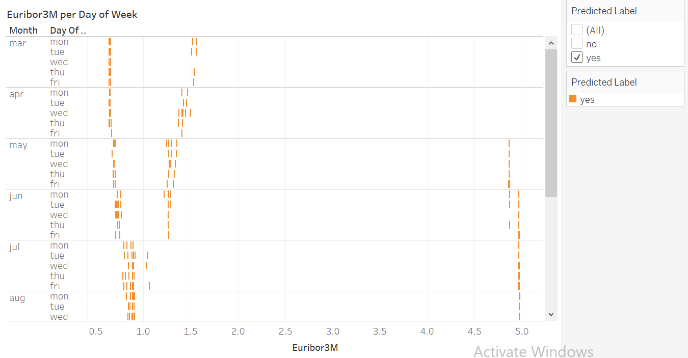


**Gambar 14. Cons Conf Idx, Cons Price Idx, Nr Employed Based on Contact**

Visualisasi tersebut menampilkan Cons Cof Idx yang merupakan indicator consumer confidence perbulan, Cons Price Idx yang merupakan indicator consumer prices perbulan, dan Nr Employed yang merupakan jumlah karyawan yang dikelompokkan berdasarkan contact dan predicted label. Diperoleh informasi bahwa Cons Price Idx dan Cons Conf Idx cellular yang lebih tinggi diprediksi tidak akan berlangganan deposito jangka panjang, begitu juga sebaliknya. Selain itu, dapat dilihat bahwa banyak karyawan yang diprediksi tidak akan berlangganan deposito dan hanya sedikit yang akan berlangganan deposito.

1. *Euribor3M per Day of Week*

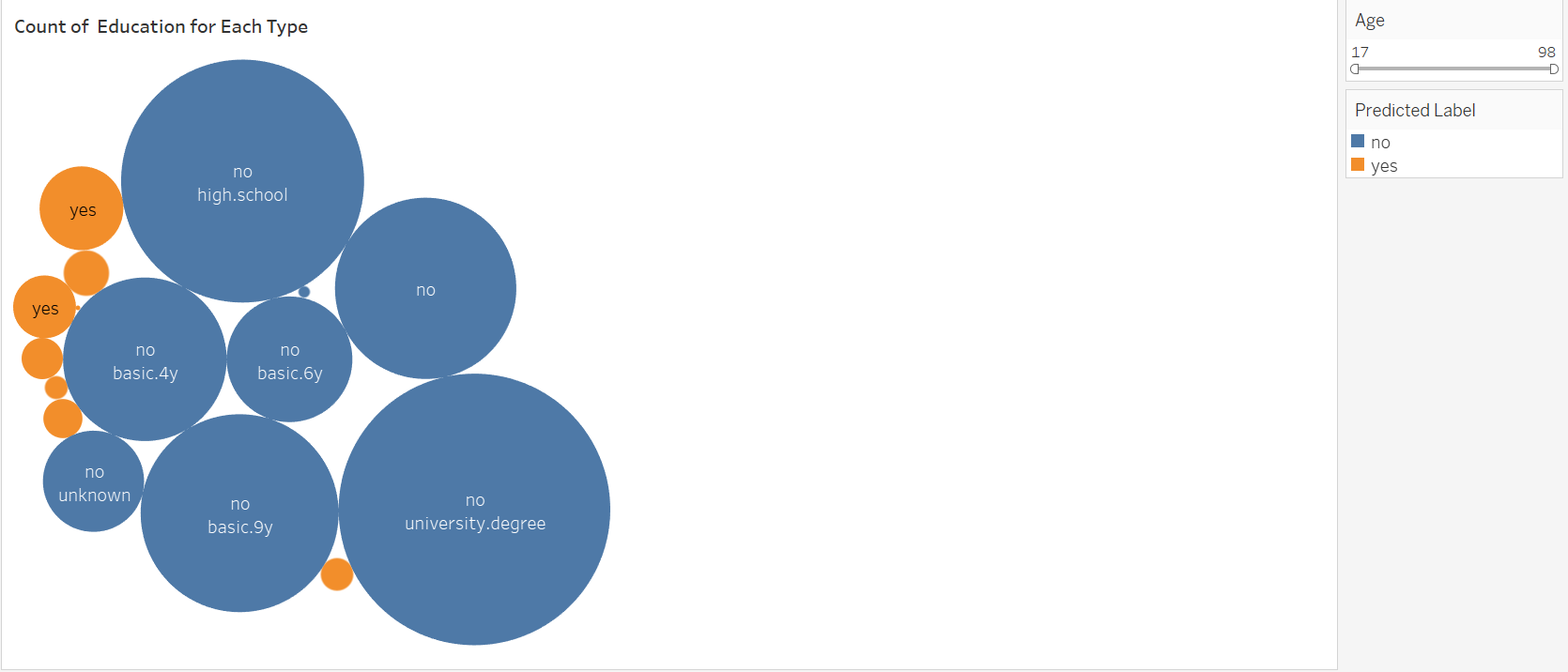


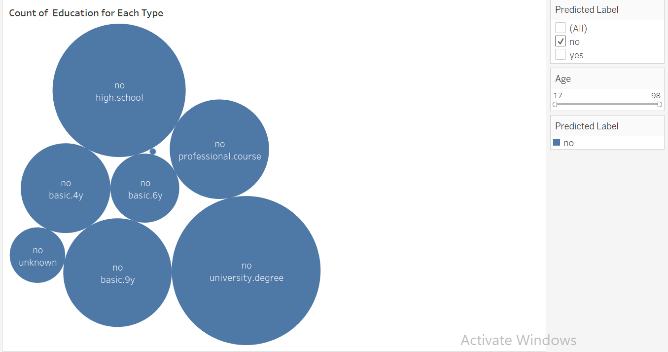
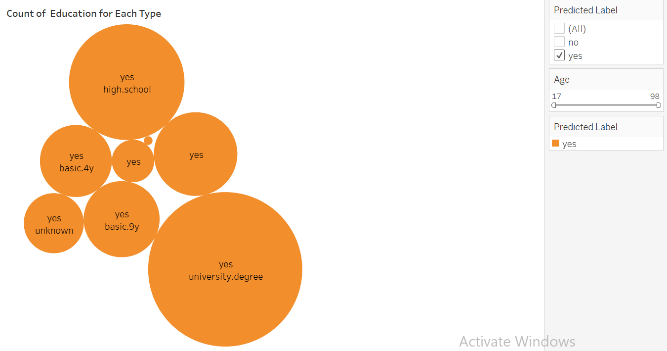
 

**Gambar 15. Euribor3M per Day of Week**

Visualisasi tersebut menampilkan Euribor3M (kurs referensi harian yang mengacu pada rata-rata suku bunga yang dipinjamkan) di setiap hari yang dikelompokkan perbulan. Prediksi label deposito jangka panjang atau tidak diberi warna biru untuk no dan warna orange untuk yes. Diperoleh informasi bahwa Euribor tidak berpengaruh terhadap kemungkinan akan melakukan deposito jangka panjang atau tidak. Hal ini dilihat bahwa sebaran label no dan yes hampir sama. Namun, pada bulan mar, oct, nop terlihat label no lebih banyak pada Euribor yang lebih tinggi dibandingkan label yes.

1. *Count of Education for Each Type*

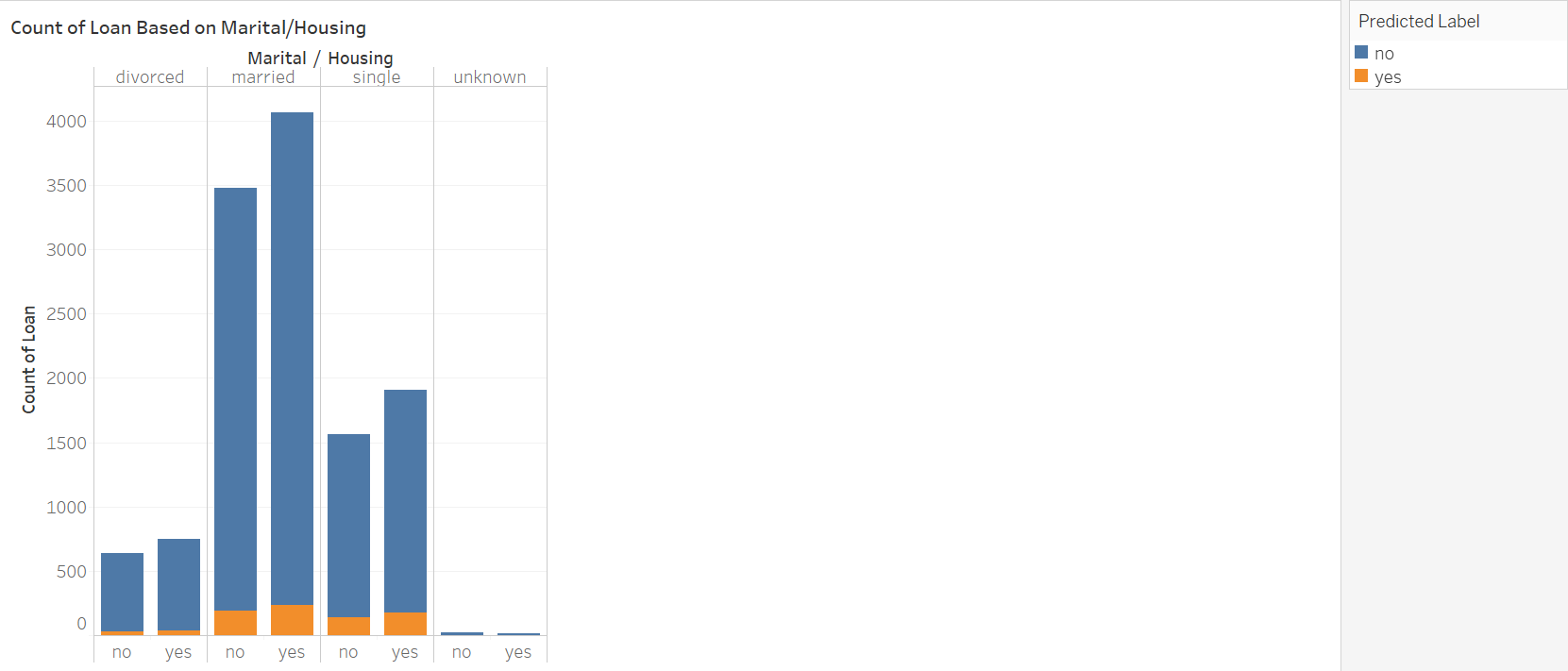


**Gambar 16. Count of Education for Each Type**

Visualisasi tersebut menampilkan jumlah dari setiap tingkat pendidikan terhadap prediksi kemungkinan akan melakukan deposito jangka panjang, “no” diberi warna biru dan “yes” diberi warna orange. Dari visualisasi dapat dilihat bahwa pendidikan university degree memiliki jumlah yang lebih banyak dibandingkan tingkat pendidikan lainnya diprediksi tidak akan melakukan deposit jangka panjang. Diikuti dengan high school, basic.9y, profesional.course, basic.4y, basic.6y, unknown, illiterate yang diprediksi juga tidak akan melakukan deposit jangka panjang. Sementara itu untuk yang diprediksi akan melakukan deposit jangka panjang dengan urutan university.degree, high.school, professional.course, basic.9y, basic.4y, unknown, basic.6y, illiterate.

1. *Count of Loan Based on Marital/Housing*



**Gambar 17. Count of Loan Based on Marital/Housing**

Visualisasi tersebut menampilkan jumlah pinjaman berdasarkan marital dan housing yang akan digunakan untuk memprediksi apakah akan melakukan deposit jangka panjang, warna biru untuk “no” dan warna orange untuk “yes”. Dapat dilihat bahwa status married dan single memiliki jumlah pinjaman yang lebih banyak dibandingkan status marital lainnya. Selain itu, marital status married dengan housing yes juga cenderung memiliki jumlah pinjaman yang lebih besar. Untuk prediksi kemungkinan akan deposito jangka panjang atau tidak, status marital married diprediksi tidak akan melakukan deposito jangka panjang dan akan melakukan deposito jangka panjang lebih banyak dibandingkan status marital lainnya. Kemudian diikuti single, divorced, dan unknown.

## **Kesimpulan**

Kesimpulan yang dapat diambil adalah algortima *Machine* *Learning* *Logistic Regression* dapat memberikan solusi terhadap prediksi dari pihak bank kepada nasabah yang akan berlangganan atau tidak, sebagai bahan evaluasi terhadap marketing bank tersebut. Klasifikasi yang dihasilkan memiliki tingkat akurasi mencapai nilai 91%, nilai ini merupakan probabilitas yang baik untuk menetukan atau memprediksi nasabah melalui *feature* (karakteristik dari nasabah).

# DAFTAR PUSTAKA

|  |  |
| --- | --- |
| [1] | T. D. H. Phan Duy Hung and T. D. Tung, "Term Deposit Subscription Prediction Using Spark MLlib and ML Packages," 25-28 February 2019. [Online]. Available: https://doi.org/10.1145/3317614.3317618. [Accessed 26 April 2021]. |
| [2] | D. PLASE, L. NIEDRITE and R. TARANOVS, "A Comparison of HDFS Compact Data Formats: Avro Versus Parquet," *MOKSLAS – LIETUVOS ATEITIS,* p. 275, 2017. |
| [3] | Q. A. Al-Radaideh and E. A. Nagi, "Using Data Mining Techniques to Build a Classification Model for Predicting Employees Performance," *(IJACSA) International Journal of Advanced Computer Science and Applications,* vol. 3, no. 2, 2012. |
| [4] | M. A. KAREEM and I. J. HUSSEIN, "The Impact of Human Resource Development on Employee Performance and Organizational Effectiveness," *Management Dynamics in the Knowledge Economy,* vol. 7, no. 3, pp. 307-322, 2019. |