Model Prediksi Churn untuk Memprediksi Churn Pelanggan

Tulisan ini bertujuan untuk memberikan pemahaman mendalam mengenai proses pengerjaan proyek Customer Churn Prediction dalam data science. Mulai dari pemahaman bisnis, pengumpulan dan eksplorasi data, hingga pengembangan model prediktif untuk memprediksi perilaku pelanggan yang cenderung meninggalkan produk atau layanan.
Churn pelanggan dapat berdampak signifikan pada pendapatan dan reputasi perusahaan. Oleh karena itu, penting untuk dapat memprediksi pelanggan yang cenderung berhenti layanan (churn) agar langkah-langkah preventif dan strategis dapat diambil guna mempertahankan pelanggan. Yang pada akhirnya membantu bisnis dalam merancang strategi proaktif untuk mempertahankan basis pelanggan mereka dan meningkatkan keberlanjutan organisasi secara keseluruhan.

Apa itu churn prediction?

Churn prediction adalah seperti menebak, tetapi tentang apakah seseorang akan berhenti menggunakan suatu produk atau layanan pada masa depan. Misalnya, bayangkan anda ingin tahu apakah teman anda akan terus berlangganan live streaming. Dengan melihat bagaimana teman anda menggunakan layanan tersebut sejauh ini (apakah sering atau jarang), anda bisa menebak apakah dia akan tetap berlangganan atau berhenti berlangganan di bulan berikutnya. Dalam bisnis, perusahaan melakukan hal serupa dengan pelanggan mereka. Mereka memeriksa pola perilaku pelanggan untuk menebak siapa yang mungkin akan berhenti menggunakan produk atau layanan mereka. Tujuannya adalah agar perusahaan dapat melakukan sesuatu untuk mencegah pelanggan tersebut berhenti, seperti membawa tawaran spesial atau pelayanan lebih baik. Ini membantu perusahaan menjaga banyak pelanggan mereka senang dan tetap menggunakan produk atau layanan yang mereka tawarkan.

Memahami model prediksi Churn

Proses prediksi churn melibatkan penggunaan model pembelajaran mesin yang bertujuan untuk memproyeksi apakah pelanggan berpotensi untuk melakukan churn. Pada tingkat konseptual, memprediksi churn pelanggan memerlukan pemahaman yang teliti mengenai profil dan perilaku pelanggan anda.

Pemahaman ini diperoleh melalui analisis data historis dari pelanggan. Kumpulan data yang efektif untuk memprediksi churn akan mencangkup sejumlah fitur yang menggambarkan pelanggan.

Selain itu, dataset juga harus memiliki variabel target yang jelas (yaitu, fitur yang ingin diprediksi). Dalam konteks ini, variabel target adalah kolom yang menandai apakah pelanggan telah melakukan churn atau tetap aktif.

Terakhir anda akan membutuhkan model machine learning, seperti algoritma Logistic Regression, Decision Tree, Random Forest, SVM, atau XGBoost, untuk menganalisis pola dalam data dan menghasilkan prediksi yang akurat.

Deskripsi Masalah:

- Churn terjadi ketika seseorang memilih untuk berhenti menggunakan layanan dari suatu bisnis. Perusahaan secara terus-menerus menganalisis fenomena churn ini karena memberikan manfaat besar bagi mereka dalam memahami pelanggan yang berpotensi meninggalkan layanan mereka.

- Tujuan dari proyek ini adalah melatih model pembelajaran mesin dengan menggunakan data yang ada. Model ini akan digunakan untuk memprediksi dengan tingkat akurasi tinggi pelanggan mana yang mungkin akan melakukan churn. hasil prediksi ini akan membantu pemilik bisnis dalam mengambil keputusan yang efektif.

Manfaat churn prediction:

1. **Peningkatan retensi pelanggan**
   Mengidentifikasi dan mempertahankan pelanggan yang berpotensi melakukan churn dapat meningkatkan tingkat retensi pelanggan, yang berdampak positif pada pendapatan jangka panjang.
2. **Efisiensi operasional**
   Memberikan wawasan yang memungkinkan perusahaan untuk mengalokasikan sumber daya dengan lebih efektif dan mengoptimalkan strategi operasional.
3. **Pengambilan keputusan yang lebih baik**
   Memberikan informasi yang diperlukan untuk mengambil keputusan bisnis yang lebih tepat dan efektif dalam rangka meminimalkan churn dan memaksimalkan pertumbuhan bisnis.
4. **Penyesuaian strategi pemasaran**
   Memungkinkan perusahaan untuk menyusun strategi pemasaran dan promosi mereka secara lebih akurat untuk mencegah churn dan mendapatkan kembali kepercayaan pelanggan.
5. **Daya saing yang lebih kuat**
   Memberikan keunggulan kompetitif dengan memungkinkan perusahaan untuk merespons perubahan pasar dengan lebih cepat dan lebih tepat, memenangkan dan mempertahankan pangsa pasar yang lebih besar.

Data Understanding

Pertama impor data dan periksa fitur - fitur nya.

```
import pandas as pd
```

```
data = pd.read_csv('train.csv')
data.head()
```

	state	account_length	area_code	international_plan	voice_mail_plan	number_vmail_messages	total_day_minutes	total_day_calls	total_day_charge	total
0	OH	107	area_code_415	no	yes	26	161.6	123	27.47	
1	NJ	137	area_code_415	no	no	0	243.4	114	41.38	
2	OH	84	area_code_408	yes	no	0	299.4	71	50.90	
3	OK	75	area_code_415	yes	no	0	166.7	113	28.34	
4	MA	121	area_code_510	no	yes	24	218.2	88	37.09	

Mari kita lihat berapa jumlah baris dan kolom di dataset yang sudah kita impor.

```
#Menampilkan jumlah baris dan kolom dataset
data.shape
```

```
(4250, 20)
```

bisa kita lihat di dalam dataset kita mempunyai 4250 baris dan 20 kolom.

Sebelum kita memulai analisis lebih lanjut, penting untuk memahami tipe data dari setiap kolom dalam dataset. Hal ini membantu kita dalam menentukan jenis transformasi dan analisis yang tepat.

```
data.dtypes
```

```
state                          object
account_length                  int64
area_code                      object
international_plan             object
voice_mail_plan                object
number_vmail_messages           int64
total_day_minutes             float64
total_day_calls                 int64
total_day_charge              float64
total_eve_minutes             float64
total_eve_calls                 int64
total_eve_charge              float64
total_night_minutes           float64
total_night_calls               int64
total_night_charge            float64
total_intl_minutes            float64
total_intl_calls                int64
total_intl_charge             float64
number_customer_service_calls   int64
churn                          object
dtype: object
```

Dalam mengawali tahap pemahaman data, perlu untuk memeriksa kualitas data yang akan digunakan dalam proyek ini. Salah satu aspek penting adalah memastikan bahwa data yang digunakan tidak mengandung nilai kosong atau Null yang dapat memengaruhi keakuratan analisis.

```
# Menampilkan jumlah nilai True pada setiap kolom dataset
print(data.isnull().sum())
```

```
state                           0
account_length                  0
area_code                       0
international_plan              0
voice_mail_plan                 0
number_vmail_messages           0
total_day_minutes               0
total_day_calls                 0
total_day_charge                0
total_eve_minutes               0
total_eve_calls                 0
total_eve_charge                0
total_night_minutes             0
total_night_calls               0
total_night_charge              0
total_intl_minutes              0
total_intl_calls                0
total_intl_charge               0
number_customer_service_calls   0
churn                           0
dtype: int64
```

Data yang kami pergunakan dalam proyek ini telah melalui proses pemrosesan awal untuk memastikan tidak ada nilai yang hilang atau Null. Hal ini penting karena keberadaan nilai Null dapat mengganggu analisis dan interpretasi data. Dengan memastikan tidak adanya nilai Null, kita dapat memastikan integritas data yang digunakan untuk mengembangkan model prediksi churn.

Kita juga akan melakukan pengecekan terhadap kemungkinan adanya data duplikat dalam dataset yang akan kita gunakan. Data duplikat adalah kejadian di mana salah satu atau lebih entri dalam dataset memiliki nilai yang sama persis di seluruh atributnya.

```
# Mengidentifikasi baris yang merupakan duplikat
duplicated_rows = data[data.duplicated()]

# Menghitung jumlah duplikat
jumlah_duplikat = len(duplicated_rows)

print("Jumlah duplikat dalam DataFrame:", jumlah_duplikat)
```

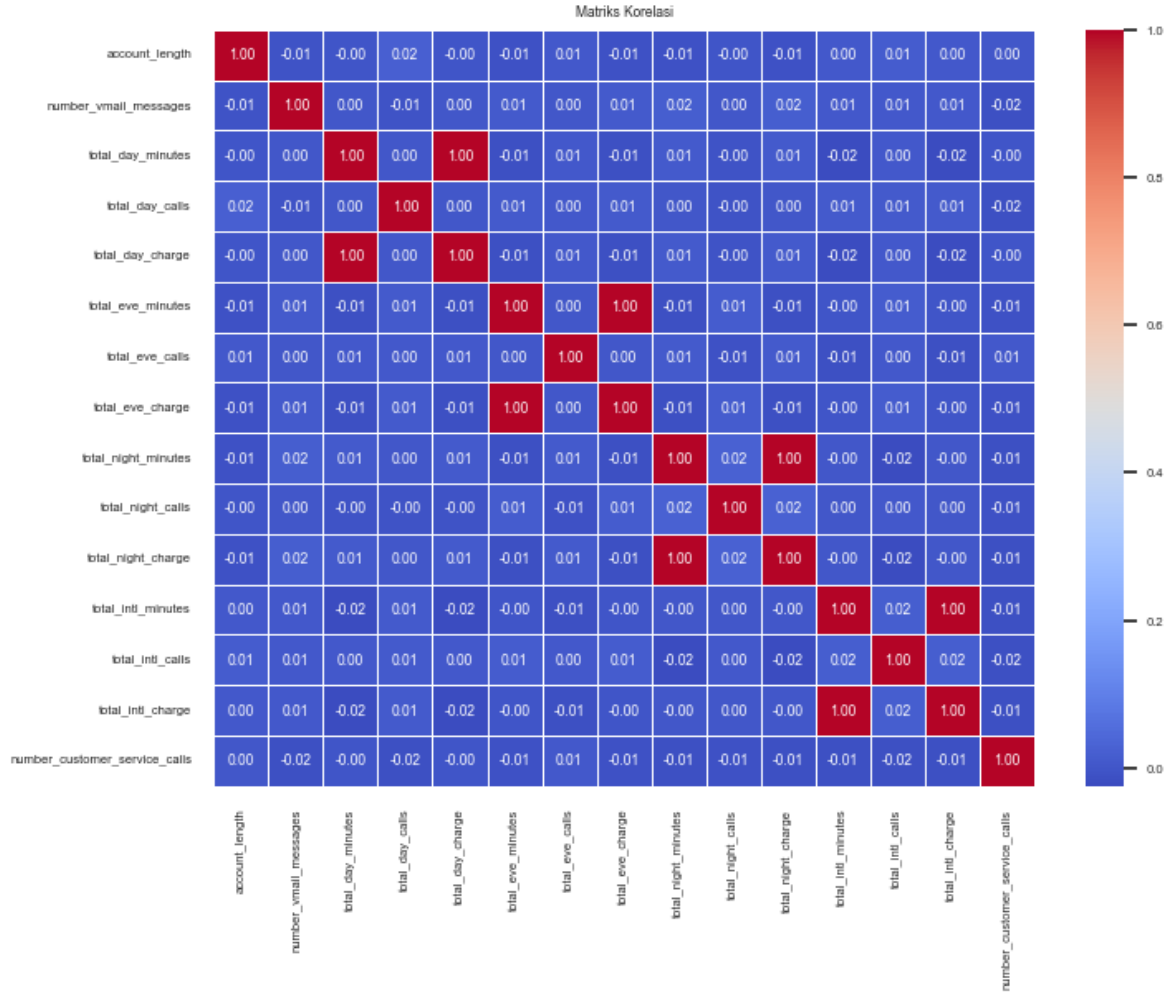
```
Jumlah duplikat dalam DataFrame: 0
```

Setelah melakukan analisis, kita memastikan bahwa tidak ada data duplikat dalam dataset ini. Keberadaan data duplikat dapat memengaruhi hasil analisis dengan memberikan bobot yang tidak tepat pada pola-pola tertentu. Dengan memastikan bahwa setiap entri dalam dataset unik, kita dapat memastikan keakuratan dan konsistensi hasil analisis yang akan kita lakukan selanjutnya.

Korelasi

Dengan menggunakan seaborn, kita dapat melihat plots matriks korelasi menggunakan heatmap. Analisis korelasi memungkinkan kita untuk mengidentifikasi sejauh mana dan bagaimana variabel - variabel saling berkaitan satu sama lain. Salah satu cara umum yang digunakan untuk memvisualisasikan korelasi adalah dengan matriks korelasi dan heatmap.

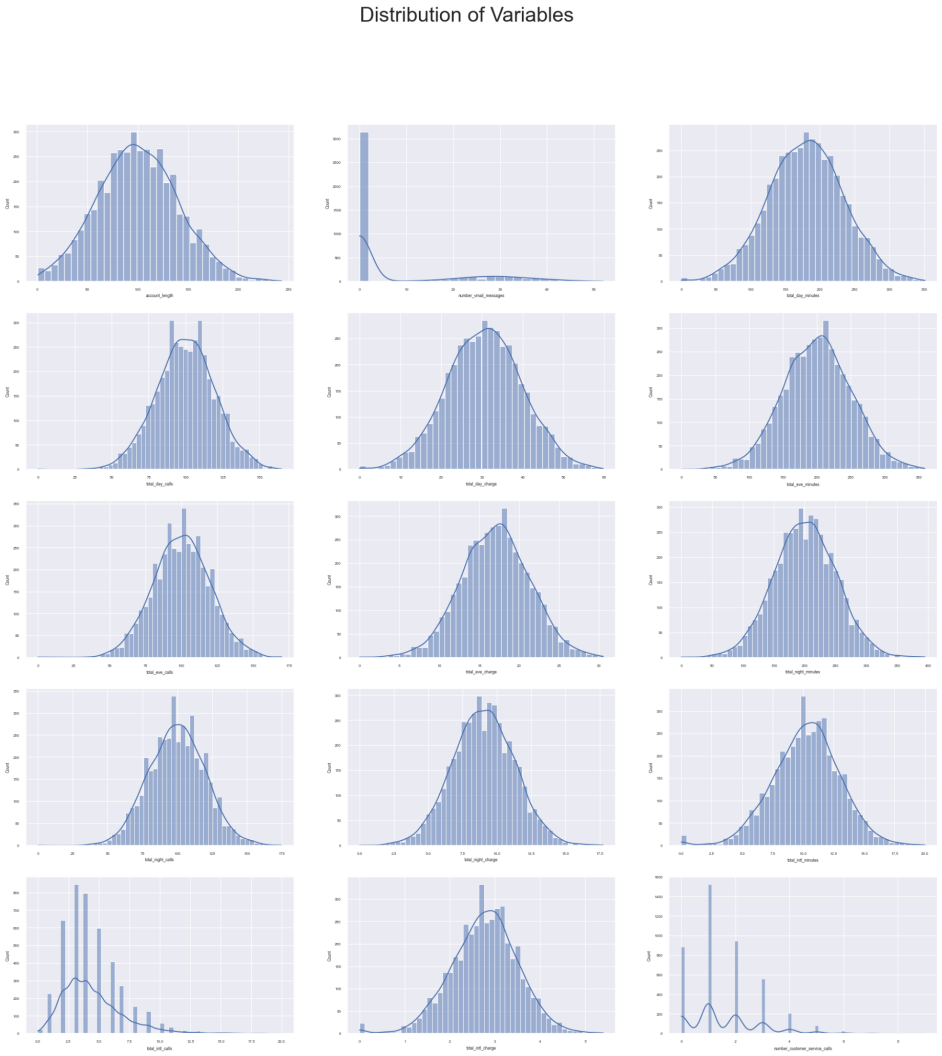
```
# Menghitung matriks korelasi
correlation_matrix = data.corr()

# Plot matriks korelasi menggunakan heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)
plt.title('Matriks Korelasi')
plt.show()
```



Koefisien korelasi antara setiap pasang variabel dalam data. Koefisien korelasi berkisar antara -1 hingga 1, di mana -1 menunjukkan korelasi negatif sempurna, 1 menunjukkan korelasi positif sempurna, dan 0 menunjukkan tidak adanya korelasi.

Distribusi fitur numerik

Selanjutnya, kita melakukan analisis terhadap distribusi dari fitur-fitur numerik yang terdapat dalam dataset. Analisis distribusi ini membantu kita untuk memahami pola, rentang nilai, dan statistik dasar dari setiap variabel numerik.

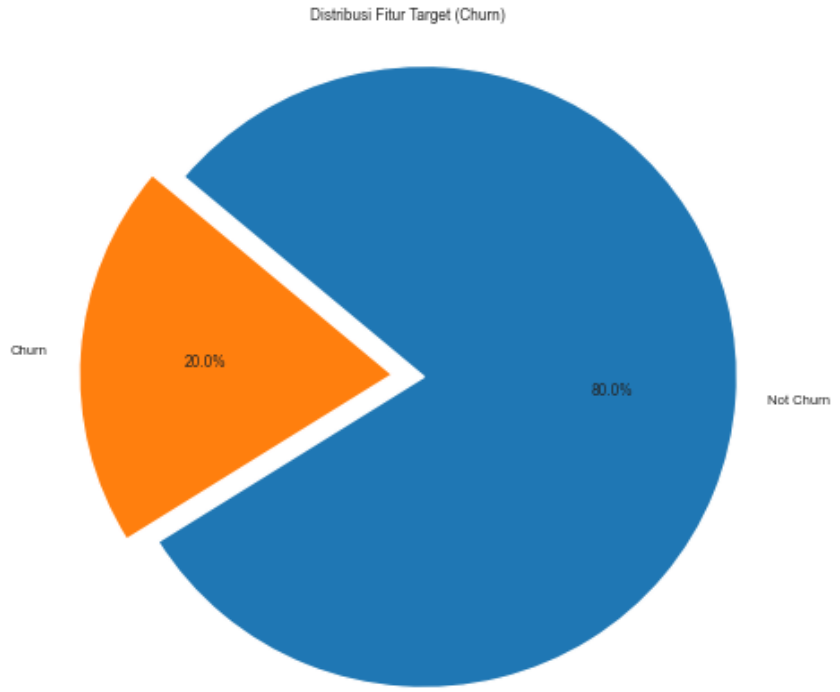


Selama analisis distribusi, ada beberapa fitur numerik menunjukkan distribusi yang condong ke kanan. Distribusi yang condong ke kanan (right-skewed) adalah distribusi yang di mana ekor panjang mengarah ke nilai-nilai yang lebih tinggi. Hal ini menunjukkan bahwa sebagian besar nilai condong pada nilai yang lebih rendah, sementara nilai-nilai tinggi memiliki frekuensi yang lebih rendah.

Distribusi yang condong ke kanan adalah informasi yang penting karena dapat memengaruhi pendekatan analisis dan pemodelan selanjutnya. Pada distribusi yang condong ke kanan, rata-rata (mean) lebih tinggi daripada median, dan sebagian besar data berada di sebelah kiri.

Pada tahap ini, kita melakukan analisis terhadap kolom target “churn” untuk memahami sejauh mana keseimbangan kelas dalam dataset. keseimbangan kelas penting untuk diperhatikan karena dapat memengaruhi kinerja dan hasil akhir dari model prediksi.

```
import matplotlib.pyplot as plt

# Data distribusi churn
labels = ['Churn', 'Not Churn']
sizes = [200, 800]  # Misalnya, 200 pengguna churn dan 800 pengguna tidak churn
colors = ['#ff7f0e','#1f77b4']  # Warna untuk setiap kategori
explode = (0.1, 0)  # Memberi efek explode pada slice "Churn"

# Membuat plot
plt.figure(figsize=(5, 5))
plt.pie(sizes, explode=explode, labels=labels, colors=colors, autopct='%1.1f%%', startangle=140)
plt.axis('equal')  # Memastikan lingkaran terlihat seperti lingkaran
plt.title('Distribusi Fitur Target (Churn)')
plt.show()
```



Setelah melakukan pemeriksaan terhadap distribusi kelas pada kolom target “churn”. Hasil analisis menunjukkan bahwa terdapat ketidakseimbangan yang signifikan antara kelas “Not Churn” dan “Churn”.

Langkah-langkah selanjutnya akan mempertimbangkan strategi yang sesuai untuk menangani ketidakseimbangan kelas, seperti oversampling, undersampling, atau menggunakan metode pembobotan yang tepat.

Data Preparation

Sebelum melanjutkan ke tahap pemodelan, kita perlu memastikan bahwa seluruh data kategorik direpresentasikan dalam bentuk yang dapat diolah oleh algoritma machine learning. Salah satu teknik yang umum digunakan untuk ini adalah label encoding.

Apa itu label encoding?
Label encoding adalah proses mengubah nilai-nilai pada variabel kategorik menjadi nilai numerik. Setiap kategori diberi label numerik unik, sehingga memungkinkan algoritma machine learning untuk memahami dan memproses informasi ini.

Misalnya, pada variabel “jenis produk”:
- “Produk A” dapat diencoding menjadi 0
- “Produk B” dapat diencoding menjadi 1
- “Produk C” dapat diencoding menjadi 2
dan seterusnya.

Implementasi label encoding
kita akan menerapkan label encoding pada variabel-variabel kategorikal dalam dataset kita sebelum melanjutkan ke tahap pemodelan. Hal ini memastikan bahwa seluruh data yang akan digunakan dalam pembangunan model telah diolah dengan benar dan siap untuk digunakan.

```
from sklearn.preprocessing import LabelEncoder

# Data kategorikal
data_kategorikal = data[['international_plan', 'voice_mail_plan', 'churn']]

# Membuat objek LabelEncoder
le = LabelEncoder()

# Melakukan Label Encoding untuk setiap kolom kategorikal
data_encoded = data_kategorikal.apply(label_encoder.fit_transform)

# Menggantikan kolom-kolom asli dengan hasil Label Encoding
data[['international_plan', 'voice_mail_plan','churn']] = data_encoded

data
```

	account_length	international_plan	voice_mail_plan	number_vmail_messages	total_day_minutes	total_day_calls	total_day_charge	total_eve_minutes	total
0	107	0	1	26	161.6	123	27.47	195.5	
1	137	0	0	0	243.4	114	41.38	121.2	
2	84	1	0	0	299.4	71	50.90	61.9	
3	75	1	0	0	166.7	113	28.34	148.3	
4	121	0	1	24	218.2	88	37.09	348.5	
...	...	...	...	...	...	...	...	...	
4245	83	0	0	0	188.3	70	32.01	243.8	
4246	73	0	0	0	177.9	89	30.24	131.2	
4247	75	0	0	0	170.7	101	29.02	193.1	
4248	50	0	1	40	235.7	127	40.07	223.0	
4249	86	0	1	34	129.4	102	22.00	267.1	

Manfaat Label Encoding
Label encoding mempermudah pemrosesan data kategorikal dalam algoritma machine learning, label encoding memungkinkan representasi yang sesuai dari data kategorikal.

Mengatasi ketidakseimbangan data dengan SMOTE
Ketidakseimbangan kelas pada kolom target “churn” adalah situasi di mana salah satu kelas memiliki frekuensi yang signifikan lebih tinggi dibandingkan kelas lainnya. Hal ini dapat memengaruhii kinerja model prediksi, khususnya ketika kelas minor memiliki informasi yang penting.

Apa itu SMOTE?
SMOTE (Synthetic Minority Over-sampling Technique) adalah metode oversampling yang digunakan untuk menangani ketidakseimbangan kelas dengan menciptakan sample sintetis untuk kelas minor. Metode ini bekerja dengan cara menghasilkan sample baru yang merupakan kombinasi linear dari tetangga terdekat dari kelas minor.

```
from sklearn.model_selection import train_test_split

# Memisahkan fitur dan target
X = data.drop('churn', axis=1)  # Fitur
y = data['churn']  # Target
```

```
# import library
from collections import Counter
from imblearn.over_sampling import SMOTE

smote = SMOTE()

# fit predictor and target variable
X_smote, y_smote = smote.fit_resample(X,y)
print('Original dataset shape', Counter(y))
print('Resample dataset shape', Counter(y_smote))
```

```
Original dataset shape Counter({0: 3652, 1: 598})
Resample dataset shape Counter({0: 3652, 1: 3652})
```

SMOTE membantu mengatasi ketidakseimbangan kelas dengan menciptakan sample sintetis, memungkinkan model untuk mendapatkan informasi lebih baik dari kelas minor dan menghasilkan prediksi yang lebih seimbang.

Pembagian data dengan ‘train-test-split’

Penting untuk membagi dataset menjadi set pelatihan (training set) dan set pengujian (testing set) sebelum melanjutkan ke tahap pemodelan. Alat yang umum digunakan untuk melakukan ini adalah fungsi ‘train_test_split’ dari scikit-learn.

‘train_test_split’ memungkinkan kita untuk membagi dataset menjadi dua bagian: satu untuk melatih model dan satu lagi untuk menguji kinerja model. Fungsinya mengambil beberapa argumen penting seperti dataset yang akan dibagi, label, ukuran set pengujian, dan pengaturan acak(random_state) untuk memastikan hasil yang dapat direproduksi.

```
# Bagi data menjadi data latih dan data uji
X_train, X_test, y_train, y_test = train_test_split(X_smote, y_smote, train_size=0.7, test_size=0.3, random_state=0)
# summarize
print('Train', X_train.shape, y_train.shape)
print('Test', X_test.shape, y_test.shape)
```

```
Train (5112, 17) (5112,)
Test (2192, 17) (2192,)
```

Dalam contoh di atas, kita memisahkan fitur dan label dari dataset. kemudian menggunakan ‘train_test_split’, data dibagi menjadi set pelatihan (80%) dan set pengujian (20%).

Mengapa penting?
Pembagian data ini penting untuk menghindari overfitting dan memastikan model yang dikembangkan memiliki kinerja yang baik pada data yang belum pernah dilihat sebelumnya.

Modeling dan Prediction

1) Logistic Regression

Logistic regression menggunakan fungsi logistik untuk menghasilkan probabilitas bahwa suatu instance atau sampel termasuk dalam kelas positif. Fungsi sigmoid memetakan nilai-nilai input ke rentang (0,1), yang dapat diinterpretasikan sebagai probabilitas.

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

lr = LogisticRegression()
lr.fit(X_train, y_train)
predictions = lr.predict(X_test)
print(f'Test accuracy: {round(accuracy_score(y_test, predictions), 4)}')
```

```
Test accuracy: 0.6747
```

Pada contoh di atas kita menggunakan implementasi Logistic Regression dari scikit-learn. Model dilatih dengan menggunakan set pelatihan (‘x_train’ dan ‘y_train’) dan kemudian digunakan untuk melakukan prediksi pada set pengujian.

2) Decision Tree Model

Decision Tree adalah algoritma klasifikasi yang membagi data menjadi subset-subset lebih kecil berdasarkan aturan keputusan yang dihasilkan. Proses ini dilakukan dengan memilih atribut yang memberikan informasi paling signifikan pada setiap tahap.

```
from sklearn.tree import DecisionTreeClassifier

clf = DecisionTreeClassifier()
clf.fit(X_train, y_train)
predictions = clf.predict(X_test)
print(f'Test accuracy: {round(accuracy_score(y_test, predictions), 4)}')
```

```
Test accuracy: 0.8672
```

Setelah melatih model Decision Tree dan melakukan evaluasi, kita mendapatkan hasil akurasi sebesar 0.8672. Artinya, model Decision Tree berhasil memprediksi dengan tingkat akurasi sekitar 86.72%

3) Random Forest Classifier

Random Forest Classifier adalah jenis model ensemble yang menggabungkan beberapa pohon keputusan untuk mencapai prediksi yang lebih baik dan lebih stabil. Setiap pohon melakukan prediksi, dan hasil akhir adalah hasil mayoritas dari semua pohon.

```python
from sklearn.ensemble import RandomForestClassifier

# Inisialisasi dan melatih model Random Forest
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train, y_train)

# Melakukan prediksi
y_pred = rf_model.predict(X_test)

# Evaluasi model
print("Accuracy:", accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.9384124087591241
```

Setelah melatih model Random Forest Classifier dan melakukan evaluasi, kita mendapatkan hasil akurasi sebesar 0.9384. Artinya, model Random Forest Classifier berhasil memprediksi dengan tingkat akurasi sekitar 93.84%

4) XGBoost

Selain model-model sebelumnya, kita juga mencoba model XGBoost sebagai opsi untuk memprediksi kemungkinan churn pada dataset.
XGBoost adalah algoritma ensemble yang sangat populer dalam machine learning. Algoritma ini menggunakan teknik boosting untuk memperkuat performa model dengan menggabungkan beberapa model lemah menjadi model kuat.

```python
import xgboost as xgb

# Inisialisasi dan latih model XGBoost
model = xgb.XGBClassifier(objective ='binary:logistic', seed=42)
model.fit(X_train, y_train)

# Lakukan prediksi pada data uji
y_pred = model.predict(X_test)

# Hitung akurasi
accuracy = accuracy_score(y_test, y_pred)
print('Akurasi:', accuracy)
```

```
Akurasi: 0.9242700729927007
```

Setelah melatih model XGBoost dan melakukan evaluasi, kita mendapatkan hasil akurasi sebesar 0.92427. Artinya, model XGBoost berhasil memprediksi dengan tingkat akurasi sekitar 92.42%

Hasil akurasi yang tinggi ini menunjukkan bahwa model XGBoost sangat efektif dalam memprediksi kemungkinan churn. Ini adalah hasil yang sangat baik dan membuat model ini menjadi pilihan yang kuat dalam pemodelan kasus ini.

Setelah mencoba beberapa model, kita memutuskan untuk memilih model XGBoost sebagai model utama untuk memprediksi kemungkinan churn pada dataset.

Model XGBoost menonjol dalam performa dan kehandalan dalam berbagai tugas machine learning, terutama dalam konteks klasifikasi. Algoritma ini dapat menangani dataset yang lebih besar, memiliki tingkat akurasi yang tinggi, dan biasanya tidak rentan terhadap overfitting.

Selain itu, XGBoost juga memiliki kemampuan untuk menangani data yang tidak seimbang (imbalanced data) dan dapat diatur secara fleksibel melalui parameter-parameter yang dapat dioptimalkan.

Evaluasi Model

1) Akurasi (Accuracy)

Akurasi adalah proposisi prediksi yang benar (baik positif maupun negatif) dari total jumlah prediksi. Nilai akurasi yang tinggi (0.924) menunjukkan bahwa model memiliki kinerja yang baik dalam memprediksi kategori churn dan non churn.

```
# Melihat akurasi model
accuracy = accuracy_score(y_test, y_pred)

print('Akurasi:', accuracy)
```

```
Akurasi: 0.9242700729927007
```

2) Presisi (Precision)

Presisi adalah proporsi positif yang benar dari keseluruhan prediksi positif. Nilai presisi yang tinggi (0.939) menunjukkan bahwa sebagian besar dari yang diprediksi sebagai churn adalah benar-benar churn.

```
#Melihat precision model
precision = precision_score(y_test, y_pred)

print('Presisi:', precision)
```

```
Presisi: 0.9385633270321361
```

3) Recall

Recall adalah proporsi prediksi positif yang benar dari keseluruhan instance yang sebenarnya positif. Nilai recall yang baik (0.908) menunjukkan bahwa model mampu menemukan sebagian besar instance yang sebenarnya positif.

```
#Melihat recall model
recall = recall_score(y_test, y_pred)

print('Recall:', recall)
```

```
Recall: 0.9076782449725777
```

4) F1 Score

F1 score adalah rata-rata harmonik dari presisi dan recall. Nilai F1 score yang tinggi (0.923) menunjukkan keseimbangan antara presisi dan recall.

```
#Melihat F1 Score
f1 = f1_score(y_test, y_pred)

print('F1 Score:', f1)
```

```
F1 Score: 0.9228624535315985
```

5) ROC AUC Score

ROC AUC Score adalah ukuran kinerja untuk model klasifikasi biner. Nilai yang tinggi (0.979) menunjukkan bahwa model memiliki kemampuan yang baik dalam membedakan antara kategori churn dan non-churn.

```
#Melihat ROC AUC Score
roc_auc = roc_auc_score(y_test, y_pred_proba)

print('ROC AUC Score:', roc_auc)
```

```
ROC AUC Score: 0.9785108706872726
```

6) Confusion Matrix

Confusion matrix adalah tabel yang mendeskripsikan performa model pada data uji, membandingkan hasil prediksi dengan nilai sebenarnya. Dalam kasus ini, terdapat 1033 prediksi true negatif (TN), 65 false positive (FP), 101 false negative (FN), dan 993 true positif (TP).

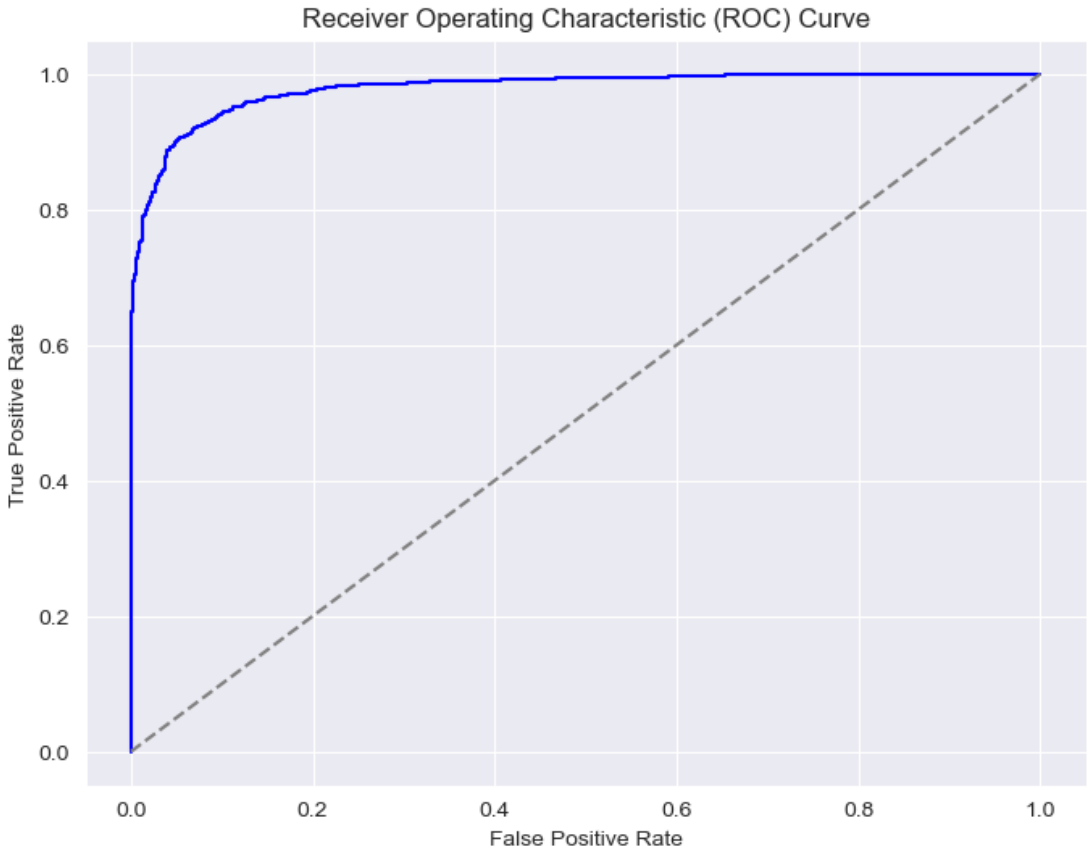
```
# Confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)
print('Confusion Matrix:')
print(conf_matrix)
```

```
Confusion Matrix:
[[1033   65]
 [ 101  993]]
```

7) Visualisasi Kurva ROC

Visualisasi kurva ROC adalah cara untuk menggambarkan kinerja sebuah model klasifikasi biner, yaitu model yang memprediksi apakah suatu objek termasuk dalam salah satu dari dua kelas. Kurva ROC menunjukkan hubungan antara tingkat positif benar (TPR) dan tingkat positif palsu (FPR) dari model pada berbagai ambang batas kalsifikasi. Makin tinggi TPR dan makin rendah FPR, makin baik model nya. Kurva ROC juga bisa digunakan untuk menghitung luas di bawah kurva (AUC), yang merupakan ukuran keseluruhan kinerja model.

```
# Visualisasi Kurva ROC
fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='b')
plt.plot([0, 1], [0, 1], color='gray', linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.show()
```

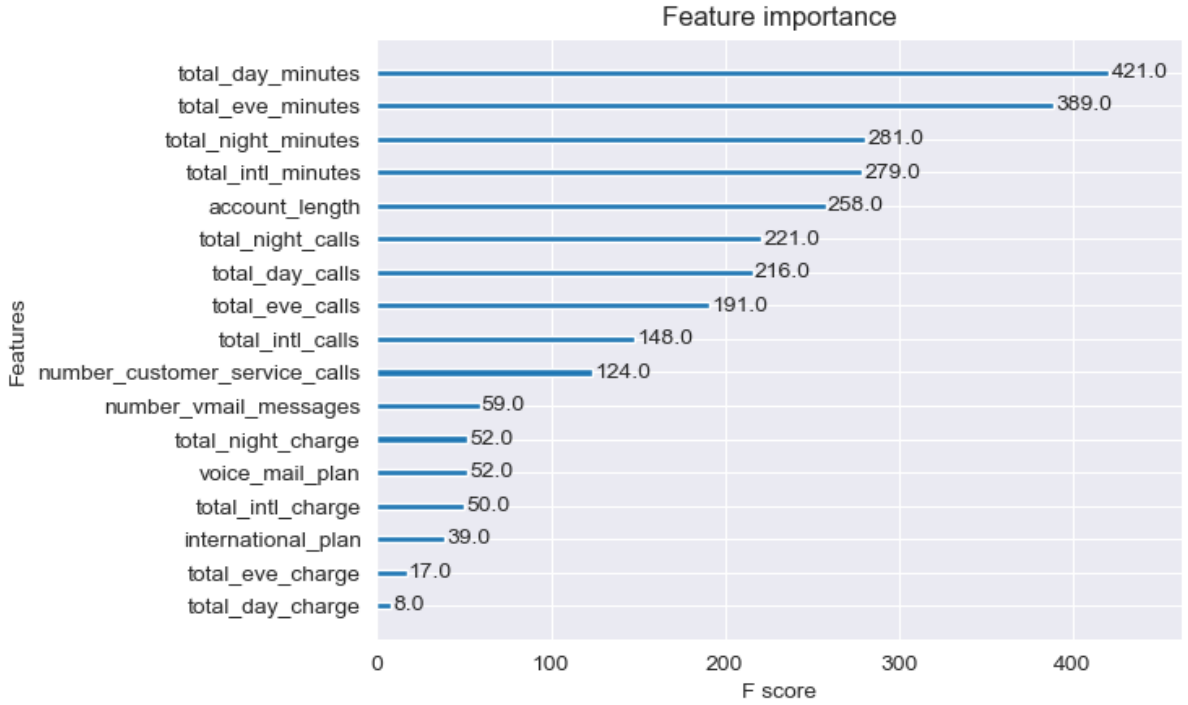


Dengan menganalisis metrik-metrik di atas, dapat disimpulkan bahwa model XGBoost memiliki kinerja yang baik dalam memprediksi churn, dengan akurasi yang tinggi dan seimbang antara presisi dan recall. Nilai ROC AUC juga menunjukkan bahwa model memiliki kemampuan diskriminatif yang kuat antara kategori churn dan non-churn.

Menampilkan tingkat penting fitur

```
# Menampilkan tingkat penting (importance) fitur
plt.figure(figsize=(10, 8))
plot_importance(model)
plt.show()
```

```
<Figure size 1000x800 with 0 Axes>
```



Tingkat penting (importance) dari fitur seperti ‘total_day_minutes’, ‘total_eve_minutes’, dan ‘total_night_minutes’ dalam konteks XGBoost dapat diinterpretasikan sebagai seberapa besar pengaruh atau kontribusi fitur tersebut terhadap proses pengambilan keputusan yang dilakukan oleh model dalam memprediksi churn.

Misalkan untuk fitur ‘total_day_minutes’ dengan tingkat penting 421.0, ini berarti bahwa fitur tersebut memiliki pengaruh yang signifikan dalam membuat prediksi churn. Jumlah menit yang digunakan dalam panggilan pada siang hari (total_day_minutes) memiliki kontribusi penting terhadap apakah pelanggan cenderung untuk beralih (churn) atau tidak.

Demikian pula, tingkat penting yang tinggi pada fitur ‘total_eve_minutes’ (389.0) dan ‘total_night_minutes’ (281.0) menunjukkan bahwa kedua fitur ini juga memiliki pengaruh besar dalam membuat prediksi churn.

Deployment

Setelah memilih model XGBoost sebagai model utama untuk memprediksi kemungkinan churn, kami akan melanjutkan dengan tahap deployment untuk membuat model siap digunakan dalam produksi.

1) Menggunakan pickle untuk serialisasi model

Salah satu cara untuk menyimpan model ke dalam file adalah dengan menggunakan modul ‘pickle’ yang memungkinkan untuk melakukan serialisasi objek python.

```python
import pickle

# Simpan model ke file menggunakan pickle
with open('xgboost_model.pkl', 'wb') as f:
    pickle.dump(model, f)

# Load model dari file
with open('xgboost_model.pkl', 'rb') as f:
    loaded_model = pickle.load(f)
```

Dalam contoh di atas, kita menggunakan ‘pickle’ untuk menyimpan model XGBoost ke dalam file dengan nama xgboost_model.pkl. Model yang sudah disimpan dapat diunggah dan digunakan kapan saja tanpa perlu melatih ulang.

2) Menggunakan joblib untuk serialisasi model

Salah satu cara untuk menyimpan model ke dalam file adalah dengan menggunakan modul ‘joblib’ yang memungkinkan untuk melakukan serialisasi objek Python. ‘joblib’ cenderung lebih cepat daripada ‘pickle’ untuk objek besar seperti model machine learning.

```python
import joblib

# Simpan model ke file menggunakan joblib
joblib.dump(model, 'xgboost_model.joblib')

# Load model dari file (gunakan salah satu, joblib atau pickle)
loaded_model = joblib.load('xgboost_model.joblib')
```

Dalam contoh di atas, kita menggunakan ‘joblib’ untuk menyimpan model XGBoost ke dalam file dengan nama xgboost_model.joblib. Model yang sudah disimpan dapat diunggah dan digunakan kapan saja tanpa perlu melatih ulang.

Kesimpulan

Setelah menerapkan model XGBoost untuk memprediksi kemungkinan churn, kita berhasil mencapai tingkat akurasi sebesar 92.42%. Hasil plot_importance menunjukan bahwa atribut-atribut seperti total_day_minutes, total_eve_minutes, dan total_night_minutes memiliki pengaruh yang signifikan dalam memprediksi kemungkinan churn, perusahaan dapat memfokuskan upaya retensi pada meningkatkan kualitas layanan dan pengalaman pelanggan pada rentang waktu ini. Hal ini memberi wawasan berharga bahwa aktifitas penggunaan layanan pada berbagai waktu hari adalah faktor krusial yang perlu diperhatikan.

Dengan memahami pola-pola perilaku pengguna yang cenderung churn, perusahaan dapat mengambil tindakan proaktif, seperti menawarkan promosi khusus, memberikan pelayanan yang lebih baik, atau menyesuaikan rencana harga untuk mempertahankan pelanggan. Langkah-langkah ini diharapkan dapat membantu mengurangi tingkat churn dan menjaga tingkat retensi pelanggan lebih tinggi, menghasilkan keuntungan yang lebih baik dan pertumbuhan bisnis yang berkelanjutan.

Referensi

IBM documentation. (n.d.). https://www.ibm.com/docs/en/spss-modeler/saas?topic=dm-crisp-help-overview

5 Teknik SMOTE untuk Overampling Data Ketidakseimbangan Anda. (2020). ICHI.PRO. https://ichi.pro/id/5-teknik-smote-untuk-overampling-data-ketidakseimbangan-anda-202401874961077

Paris, R. (2022, May 14). How to handle an imbalanced dataset? - Rohan Paris - Medium. Medium. https://parisrohan.medium.com/how-to-handle-an-imbalanced-dataset-9b9012f07017

Team, U. (2023, September 20). How to build a churn prediction model to predict customer churn. Medium. https://userpilot.medium.com/how-to-build-a-churn-prediction-model-to-predict-customer-churn-8c16da37198c

Cinthya. (2022, December 16). Customer Churn: Pengertian, Cara Hitung, dan Cara Menanggulanginya. Accurate Online. https://accurate.id/marketing-manajemen/customer-churn/

Selvaraj, N. (2022, September 16). 8 machine learning models explained in 20 minutes. https://www.datacamp.com/blog/machine-learning-models-explained