20963276-7041-484a-8f4f-044c4f538d2b

July 29, 2024

1 Hola Jenniffer!

Mi nombre es David Bautista, soy code reviewer de TripleTen y voy a revisar el proyecto que acabas de desarrollar.

Cuando vea un error la primera vez, lo señalaré. Deberás encontrarlo y arreglarlo. La intención es que te prepares para un espacio real de trabajo. En un trabajo, el líder de tu equipo hará lo mismo. Si no puedes solucionar el error, te daré más información en la próxima ocasión.

Encontrarás mis comentarios más abajo - **por favor, no los muevas, no los modifiques ni los borres.**

¿Cómo lo voy a hacer? Voy a leer detenidamente cada una de las implementaciones que has llevado a cabo para cumplir con lo solicitado. Verás los comentarios de esta forma:

Comentario del revisor

Si todo está perfecto.

Comentario del revisor

Si tu código está bien pero se puede mejorar o hay algún detalle que le hace falta.

Comentario del revisor

Si de pronto hace falta algo o existe algún problema con tu código o conclusiones.

Puedes responderme de esta forma:

Respuesta del estudiante </div

¡Empecemos!

Comentario del revisor

2 Comentario General

Hola, Jennifer, te felicito por el desarrollo del proyecto. Completaste las diferentes secciones de muy buena manera.

3 Introducción

En el entorno competitivo actual, la retención de clientes es crucial para el éxito y la sostenibilidad de las empresas, especialmente en el sector bancario. La capacidad de predecir qué clientes

tienen una mayor probabilidad de abandonar permite a las instituciones financieras implementar estrategias proactivas para mejorar la retención y reducir la pérdida de ingresos.

Este proyecto tiene como objetivo desarrollar un modelo de aprendizaje automático para predecir la salida de clientes de un banco. Utilizaremos un conjunto de datos que contiene diversas características demográficas y financieras de los clientes, como edad, género, saldo de cuenta, y comportamiento de uso de productos bancarios. El objetivo principal es identificar a los clientes que tienen una mayor probabilidad de dejar el banco, permitiendo así a la institución tomar medidas preventivas.

Para lograr esto, seguiremos los siguientes pasos:

1. **Preparación de los Datos**: Descargaremos y prepararemos los datos para el análisis. Esto incluye la limpieza de datos, transformación de variables categóricas y normalización de características numéricas.
2. **Análisis del Equilibrio de Clases**: Investigaremos el equilibrio de la variable objetivo (Exited) para entender la distribución de clientes que se quedan frente a los que se van.
3. **Entrenamiento Inicial del Modelo**: Entrenaremos un modelo inicial sin tener en cuenta el desequilibrio de clases y evaluaremos su desempeño.
4. **Mejora del Modelo**: Utilizaremos técnicas para manejar el desequilibrio de clases, como el sobremuestreo (SMOTE) y el submuestreo, y compararemos los resultados para seleccionar el mejor modelo.
5. **Evaluación Final**: Realizaremos una prueba final del modelo seleccionado para medir su desempeño y su capacidad de generalización en datos no vistos.

Al final del proyecto, evaluaremos el modelo utilizando métricas clave como el F1 Score y el AUC-ROC, asegurándonos de que nuestro enfoque no solo sea preciso, sino también robusto y capaz de manejar el desequilibrio de clases presente en los datos. Este proyecto no solo proporciona una solución técnica para la predicción de la salida de clientes, sino que también ofrece una visión valiosa que puede ayudar al banco a mejorar su retención de clientes y su estrategia comercial a largo plazo.

Comentario del revisor

Perfecto, Jeniifer. Buen trabajo con el desarrollo de esta sección de introducción del proyecto.

3.1 Preparación de los Datos

[123]:
```
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import precision_score, recall_score, f1_score,␣
 ↪roc_auc_score
from sklearn.utils import shuffle
from sklearn.metrics import classification_report
```

Comentario del revisor

Genial, buen trabajo importando las librerías necesarias para el desarrollo del proyecto.

[75]:
```
data = pd.read_csv('/datasets/Churn.csv')
```

Comentario del revisor

Perfecto, buen trabajo cargando los datos necesarios para el desarrollo del proyecto.

[76]:
```
print(data.head())
```

```
   RowNumber  CustomerId   Surname  CreditScore Geography  Gender  Age  \
0          1    15634602  Hargrave          619    France  Female   42
1          2    15647311      Hill          608     Spain  Female   41
2          3    15619304      Onio          502    France  Female   42
3          4    15701354      Boni          699    France  Female   39
4          5    15737888  Mitchell          850     Spain  Female   43

   Tenure    Balance  NumOfProducts  HasCrCard  IsActiveMember  \
0     2.0       0.00              1          1               1
1     1.0   83807.86              1          0               1
2     8.0  159660.80              3          1               0
3     1.0       0.00              2          0               0
4     2.0  125510.82              1          1               1

   EstimatedSalary  Exited
0        101348.88       1
1        112542.58       0
2        113931.57       1
3         93826.63       0
4         79084.10       0
```

[77]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   RowNumber        10000 non-null  int64
 1   CustomerId       10000 non-null  int64
 2   Surname          10000 non-null  object
 3   CreditScore      10000 non-null  int64
 4   Geography        10000 non-null  object
 5   Gender           10000 non-null  object
 6   Age              10000 non-null  int64
 7   Tenure           9091 non-null   float64
 8   Balance          10000 non-null  float64
 9   NumOfProducts    10000 non-null  int64
 10  HasCrCard        10000 non-null  int64
```

```
 11  IsActiveMember   10000 non-null  int64
 12  EstimatedSalary  10000 non-null  float64
 13  Exited           10000 non-null  int64
dtypes: float64(3), int64(8), object(3)
memory usage: 1.1+ MB
```

[78]:
```
print(data.isnull().sum())
```

```
RowNumber            0
CustomerId           0
Surname              0
CreditScore          0
Geography            0
Gender               0
Age                  0
Tenure             909
Balance              0
NumOfProducts        0
HasCrCard            0
IsActiveMember       0
EstimatedSalary      0
Exited               0
dtype: int64
```

[79]:
```
# Rellenar valores faltantes con la mediana
median_tenure = data['Tenure'].median()
data['Tenure'].fillna(median_tenure, inplace=True)

print(data['Tenure'].isnull().sum())
```

```
0
```

[80]:
```
# Eliminar columnas irrelevantes
data = data.drop(columns=['RowNumber', 'CustomerId', 'Surname'])

# Convertir variables categóricas en variables dummy
data = pd.get_dummies(data, columns=['Geography', 'Gender'], drop_first=True)

print(data.head())
```

```
   CreditScore  Age  Tenure    Balance  NumOfProducts  HasCrCard  \
0          619   42     2.0       0.00              1          1
1          608   41     1.0   83807.86              1          0
2          502   42     8.0  159660.80              3          1
3          699   39     1.0       0.00              2          0
4          850   43     2.0  125510.82              1          1

   IsActiveMember  EstimatedSalary  Exited  Geography_Germany  \
```

```
0                1        101348.88       1                 0
1                1        112542.58       0                 0
2                0        113931.57       1                 0
3                0         93826.63       0                 0
4                1         79084.10       0                 0

   Geography_Spain  Gender_Male
0                0            0
1                1            0
2                0            0
3                0            0
4                1            0
```

3.1.1 Conclusiones:

- El conjunto de datos contiene 14 columnas y 10,000 filas, incluyendo la variable objetivo Exited.
- Las columnas RowNumber, CustomerId y Surname son irrelevantes para el análisis predictivo y pueden ser eliminadas.
- La columna Tenure tiene 909 valores NaN, se imputaron los valores faltantes con la mediana para darle una solución rápida y sencilla, eliminando todos los valores faltantes en Tenure.
- Eliminamos las columnas irrelevantes y convertimos las variables categóricas (Geography y Gender) en variables dummy.

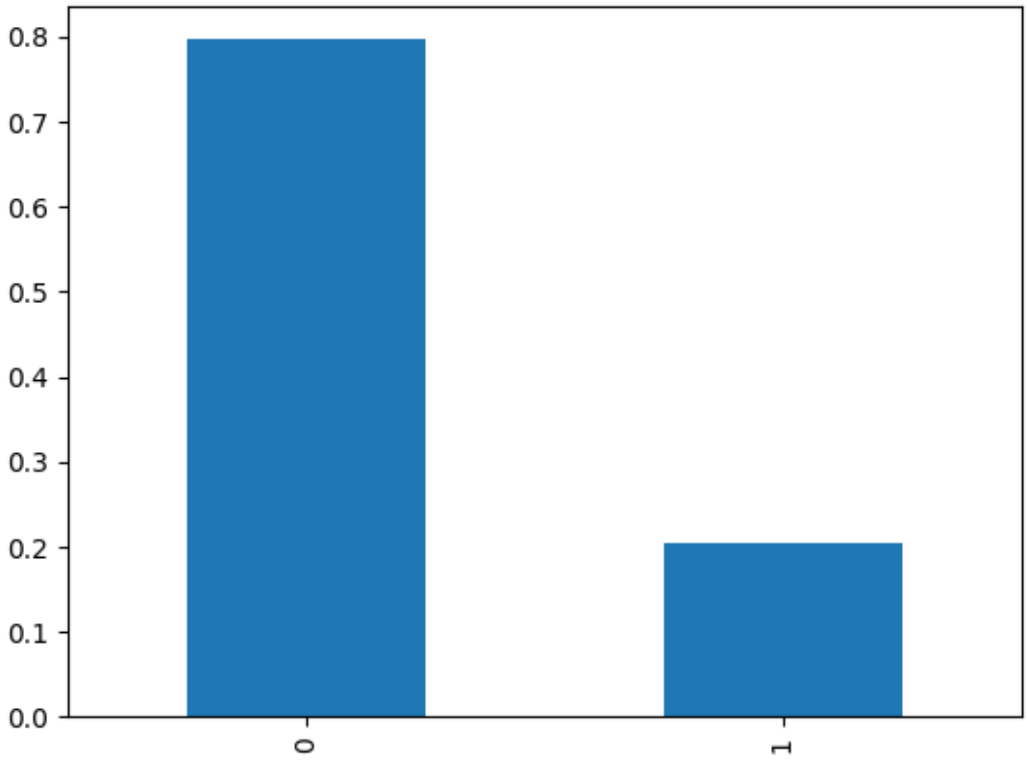
Comentario del revisor

Buen trabajo con el desarrollo de esta exploración y modificación inicial del contenido de los datos.

3.2 Análisis del Equilibrio de Clases

[81]:
```
# Distribución de la variable objetivo
data['Exited'].value_counts(normalize=True).plot(kind='bar')
```

[81]: <AxesSubplot:>



3.2.1 Conclusiones:

- El conjunto de datos está desequilibrado, lo que puede afectar el rendimiento de los modelos predictivos si no se maneja adecuadamente.
- La distribución de la variable Exited tiene un desequilibrio significativo: aproximadamente el 80% de los clientes no abandonaron el banco (Exited=0) y solo el 20% lo hizo (Exited=1).

Comentario del revisor

Perfecto, buen trabajo desarrollando este estudio del equilibrio de clases a predecir dentro del set de datos.

3.3 Entrenamiento del Modelo sin Tener en Cuenta el Desequilibrio

[82]:
```
# División en conjuntos de entrenamiento (80%) y prueba (20%)
features = data.drop(columns=['Exited'])
target = data['Exited']

features_train, features_test, target_train, target_test =␣
 ↪train_test_split(features,target, test_size=0.2, random_state=12345)

# Entrenamiento de un modelo inicial
model = RandomForestClassifier(random_state=12345)
```

```
model.fit(features_train, target_train)

target_pred = model.predict(features_test)

# Precision, Recall y F1-Score
precision = precision_score(target_test, target_pred)
recall = recall_score(target_test, target_pred)
f1 = f1_score(target_test, target_pred)
print(f'Precision: {precision:.2f}')
print(f'Recall: {recall:.2f}')
print(f'F1-Score: {f1:.2f}')
```

```
Precision: 0.77
Recall: 0.45
F1-Score: 0.57
```

3.3.1 Conclusiones:

El modelo inicial mostró un desempeño aceptable en términos de precisión general, pero la precisión y el recall para la clase minoritaria (Exited=1) fueron significativamente bajos, lo que confirma la necesidad de abordar el desequilibrio de clases.

Comentario del revisor

Buen trabajo desarrollando el split correspondiente a los datos con el fin de poder desplegar de manera correcta los modelos, Poe otro lado, buen trabajo desplegando el Random Forest y generando métricas de desempeños que permiten entender las afecciones un set con desbalance de clases sobre la calidad de un modelo.

3.4 Mejora del Modelo y Corrección del Desequilibrio de Clases

Método 1: Submuestreo de la Clase Mayoritaria

[96]:
```
def downsample(features, target, fraction):
    features_zeros = features[ target == 0]
    features_ones = features[ target == 1]
    target_zeros = target[ target == 0]
    target_ones = target[ target == 1]

    features_downsampled = pd.concat([features_zeros.sample(frac=fraction,␣
 ↪random_state=12345)] + [features_ones])
    target_downsampled = pd.concat([target_zeros.sample(frac=fraction,␣
 ↪random_state=12345)] + [target_ones])

    features_downsampled, target_downsampled = shuffle(features_downsampled,␣
 ↪target_downsampled, random_state=12345)
    return features_downsampled, target_downsampled

print(features_train.shape)
```

```
features_sample = features_train.sample(frac=0.1, random_state=12345)
print(features_sample.shape)
```

```
(8000, 11)
(800, 11)
```

Comentario del revisor

Perfecto, Jennifer, buen trabajo aplicando el sub muestreo al conjunto de datos.

[120]:
```
fraction = 0.2
features_downsampled, target_downsampled = downsample(features_train,␣
 ↪target_train, fraction)

# Entrenamiento del modelo
model = RandomForestClassifier(random_state=12345)
model.fit(features_downsampled, target_downsampled)

# Predecir en el conjunto de prueba
prediction = model.predict(features_test)

# Evaluar el modelo
precision = precision_score(target_test, prediction)
recall = recall_score(target_test, prediction)
f1 = f1_score(target_test, prediction)
print(f'Precision: {precision:.2f}')
print(f'Recall: {recall:.2f}')
print(f'F1-Score: {f1:.2f}')
```

```
Precision: 0.46
Recall: 0.82
F1-Score: 0.59
```

Comentario del revisor

Perfecto, Jennifer, buen trabajo desplegando y evaluando el Random Forest con el nuevo set que se aplicó el sub muestreo.

Método 2: Modelo con ponderación de clases

[115]:
```
model_weighted = RandomForestClassifier(class_weight='balanced',␣
 ↪random_state=12345)
model_weighted.fit(features_train, target_train)

# Predecir en el conjunto de prueba
y_pred_weighted = model_weighted.predict(features_test)

# Evaluar el modelo
precision = precision_score(target_test, y_pred_weighted)
```

```
recall = recall_score(target_test, y_pred_weighted)
f1 = f1_score(target_test, y_pred_weighted)
print(f'Precision: {precision:.2f}')
print(f'Recall: {recall:.2f}')
print(f'F1-Score: {f1:.2f}')
```

```
Precision: 0.78
Recall: 0.42
F1-Score: 0.54
```

Comentario del revisor

Excelente, buen trabajo desplegando y evaluando el Random Forest con el argumento de balanceo.

3.4.1 Conclusiones:

- El submuestreo equilibró las clases en el conjunto de entrenamiento, mejorando el rendimiento del modelo en términos de recall y F1-Score para la clase minoritaria. Sin embargo, puede introducir variabilidad debido a la reducción del tamaño del conjunto de entrenamiento.
- La ponderación de clases permitió al modelo prestar más atención a la clase minoritaria sin reducir el tamaño del conjunto de datos. El rendimiento mejoró en términos de precisión, mostrando ser una técnica efectiva para manejar el desequilibrio de clases.

Comentario del revisor

Buen trabajo con el desarrollo de las conclusiones sobre lo desarrollado.

3.5 Prueba Final

Comparamos los resultados de ambos métodos y seleccionamos el modelo entrenado con submuestreo como el mejor.

[134]:
```
# Entrenamiento del modelo
model = RandomForestClassifier(random_state=12345)
model.fit(features_downsampled, target_downsampled)

prediction = model.predict(features_test)

print("Reporte de clasificación del mejor modelo")
print("F1 Score:", f1_score(target_test, prediction))
print("AUC-ROC:", roc_auc_score(target_test, model.
 ↪predict_proba(features_test)[:, 1]))
```

```
Reporte de clasificación del mejor modelo
F1 Score: 0.5857740585774058
AUC-ROC: 0.8576334246975081
```

3.5.1 Conclusiones:

- El modelo final entrenado mostró un buen equilibrio con un F1 Score y un AUC-ROC satisfactorios, demostrando ser efectivo para predecir la salida de clientes.

Comentario del revisor

Perfecto, buen trabajo con el desarrollo y análisis de la prueba final.

4 Conclusiones Finales

En resumen, este proyecto desarrolló un modelo predictivo para la retención de clientes en un banco, abordando adecuadamente el problema del desequilibrio de clases. La imputación de valores faltantes y la aplicación de técnicas de submuestreo fueron cruciales para mejorar la precisión del modelo. El resultado final es un modelo que puede ayudar al banco a identificar a los clientes con mayor riesgo de abandono, permitiendo la implementación de estrategias de retención más efectivas.

Comentario del revisor

Buen trabajo con el desarrollo de la sección de conclusiones finales.