```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

```
df = pd.read_csv('/content/churn_cleaned.csv')
```

```
df.head()
```

```
   RowNumber  CustomerId   Surname  CreditScore  Geography  Gender  Age  Tenure    Balance  NumOfPr
0          1    15634602  Hargrave          619     France  Female   42       2       0.00
1          2    15647311      Hill          608      Spain  Female   41       1   83807.86
2          3    15619304      Onio          502     France  Female   42       8  159660.80
3          4    15701354      Boni          699     France  Female   39       1       0.00
4          5    15737888  Mitchell          850      Spain  Female   43       2  125510.82
```

Next steps: Generate code with df New interactive sheet

```
df.tail()
```

```
      RowNumber  CustomerId    Surname  CreditScore  Geography  Gender  Age  Tenure    Balance  Num
9995       9996    15606229   Obijiaku          771     France    Male   39       5       0.00
9996       9997    15569892  Johnstone          516     France    Male   35      10   57369.61
9997       9998    15584532        Liu          709     France  Female   36       7       0.00
9998       9999    15682355  Sabbatini          772    Germany    Male   42       3   75075.31
9999      10000    15628319     Walker          792     France  Female   28       4  130142.79
```

```
# I want to encode categorical colums that dont have numbers in them or groups like the ones
#in age group
df = pd.get_dummies(df, columns=['Geography', 'Gender', 'customer_age_group'], drop_first=True)
```

```
# im just going to drop the name column as it doesnt do anything to train the model later on
#error is because i ran it twice
df = df.drop(columns=['Surname'])
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
/tmp/ipython-input-369810290.py in <cell line: 0>()
      1 # im just going to drop the name column as it doesnt do anything to train the model later
on
----> 2 df = df.drop(columns=['Surname'])

                                ⌃ 3 frames
/usr/local/lib/python3.12/dist-packages/pandas/core/indexes/base.py in drop(self, labels, errors)
   7068         if mask.any():
   7069             if errors != "ignore":
-> 7070                 raise KeyError(f"{labels[mask].tolist()} not found in axis")
   7071             indexer = indexer[~mask]
   7072         return self.delete(indexer)

KeyError: "['Surname'] not found in axis"
```

Next steps: Explain error

```
df.head()
```

Next steps: Generate code with df New interactive sheet

```
# so i forgot to hot encode the balance category so ill do that now
df = pd.get_dummies(df, columns=['balance_category'], drop_first=True)
```

```
df.head()
```

Next steps: Generate code with df New interactive sheet

```
# so now im going to split the features into x and y
#x being the features used to predict churn, so everything but churn
#and y is just churn

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

```
# so im going to do a test/train split
# 30% of the data is for testing the rest for training
# the random state and stratify is to ensure results arent skewed
#and also is reproducable

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42, stratify=y
)
```

```
# now im going to scale it since the number difference for things like balance
# can be quite large

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
# now all the preproccessing is done, now we train the model
# so i went with the logistic regression model because it predicts the probability
# of the relationship between churn and the rest of the variables
# i set the max interation to 1000 as it is a larger dataset

log_reg = LogisticRegression(max_iter=1000)
log_reg.fit(X_train_scaled, y_train)
```

```
▾    LogisticRegression      ⓘ ?
LogisticRegression(max_iter=1000)
```

```
# now i just set the predicted class and then find the probability for each class
#the [:, 1] is because we only want the probabilty of churn, as it spits out
# 2 columns one being probability of not churn which we dont want

y_pred = log_reg.predict(X_test_scaled)
y_prob = log_reg.predict_proba(X_test_scaled)[:, 1]
```

```
# now that the model is trained its time to identify key churn drivers
#essentially what i did here is grab the coefficients and then combine feature
# name to the said coefficients

importance = log_reg.coef_[0]

feature_importance = pd.DataFrame({
    'Feature': X.columns,
    'Coefficient': importance
}).sort_values(by='Coefficient', ascending=False)
```

```
feature_importance
```

	Feature	Coefficient
3	Age	0.645530
14	customer_age_group_45-59	0.370491
10	Geography_Germany	0.322664
5	Balance	0.096545
17	balance_category_Medium	0.063611
9	EstimatedSalary	0.050422
4	Tenure	-0.000258
11	Geography_Spain	-0.006137
16	balance_category_Low	-0.006907
7	HasCrCard	-0.008434
13	customer_age_group_30-44	-0.009830
6	NumOfProducts	-0.018319
1	CustomerId	-0.023797
18	balance_category_Zero	-0.032511
0	RowNumber	-0.041755
2	CreditScore	-0.086455
15	customer_age_group_60+	-0.172485
12	Gender_Male	-0.238606
8	IsActiveMember	-0.481318

Next steps: Generate code with `feature_importance` New interactive sheet

```
# just from looking at this we can see that age comes into play again
# as a positive coefficent is a increase in churn risk and a lower is a decrease
# so a higher age increases churn risk, which is exactly what we hypothesised before
```

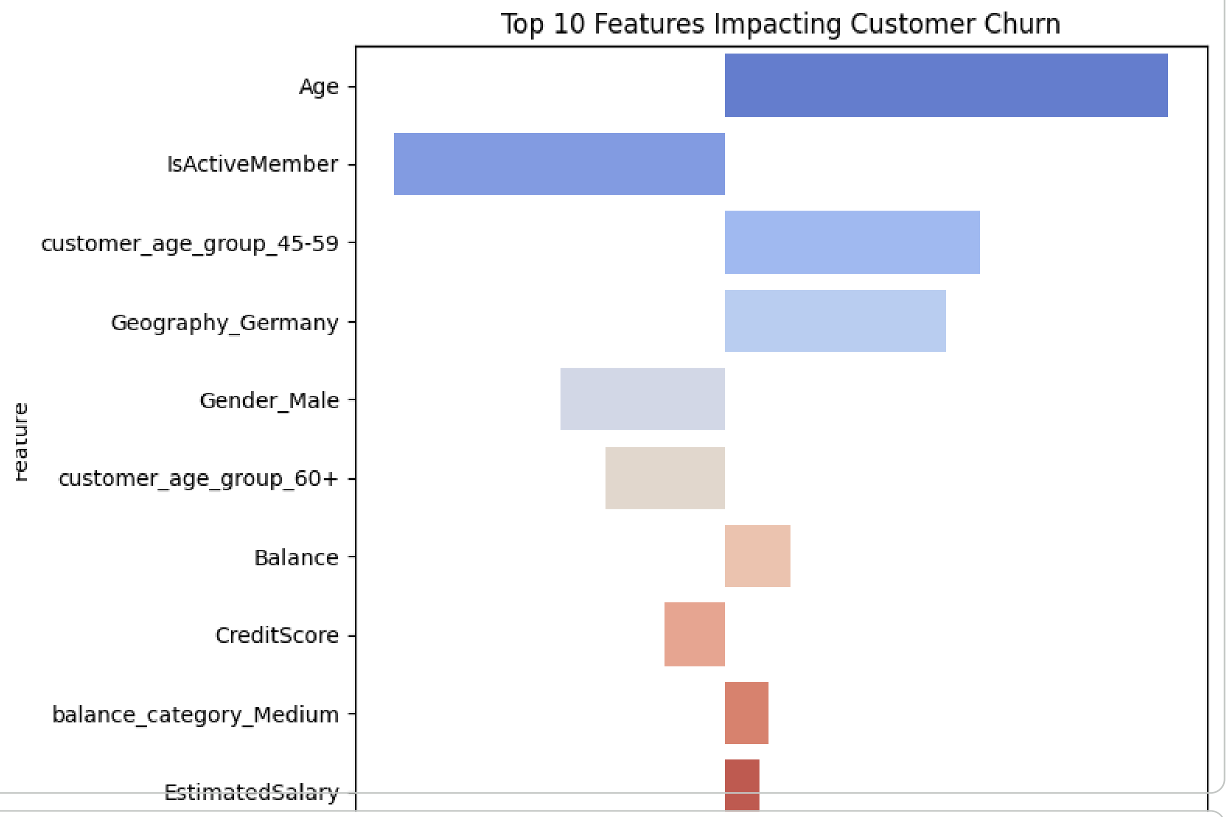
```
# i think it is also useful to try visualise it

plt.figure(figsize=(8,6))
sns.barplot(
    y='Feature',
    x='Coefficient',
    data=feature_importance.sort_values('Coefficient', key=abs, ascending=False).head(10),
    palette='coolwarm'
)
plt.title('Top 10 Features Impacting Customer Churn')
plt.xlabel('Coefficient Value (Impact on Churn)')
plt.ylabel('Feature')
plt.tight_layout()
plt.show()
```

```
:mp/ipython-input-2142083005.py:4: FutureWarning:

ıssing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the

 sns.barplot(
```



```
# so from here on top of age being a big reason for customer churn, we see that there
#is something wrong with germany as well, we mentioned this in our previous analysis as well
# so there obviously must be some product or service issue with the german bank branch
```