Part 3 Data Analysis

Cassie Kamens and Matt Werner

BAN/DAT Project

Credit Card Customer Turnover

Import statements and datasets- the full dataset as well as the test and train

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

```
from google.colab import files
uploaded = files.upload()
```

Choose Files 3 files

- **BankChurners.csv**(text/csv) - 1394543 bytes, last modified: 2/17/2023 - 100% done
- **Churn_test_set.csv**(text/csv) - 266706 bytes, last modified: 2/15/2023 - 100% done
- **Churn_training_set.csv**(text/csv) - 1066131 bytes, last modified: 2/15/2023 - 100% done

```
Saving BankChurners.csv to BankChurners.csv
Saving Churn_test_set.csv to Churn_test_set.csv
Saving Churn_training_set.csv to Churn_training_set.csv
```

```
full_file="BankChurners.csv"

train_infile="Churn_training_set.csv"

test_infile="Churn_test_set.csv"

full_df=pd.read_csv(full_file)

train_df=pd.read_csv(train_infile)

test_df=pd.read_csv(test_infile)
```

Explore the shape and contents of the data

```
test_df.columns[0:21]
```

```
Index(['CLIENTNUM', 'Attrition_Flag', 'Customer_Age', 'Gender',
       'Dependent_count', 'Education_Level', 'Marital_Status',
       'Income_Category', 'Card_Category', 'Months_on_book',
       'Total_Relationship_Count', 'Months_Inactive_12_mon',
       'Contacts_Count_12_mon', 'Credit_Limit', 'Total_Revolving_Bal',
       'Avg_Open_To_Buy', 'Total_Amt_Chng_Q4_Q1', 'Total_Trans_Amt',
       'Total_Trans_Ct', 'Total_Ct_Chng_Q4_Q1', 'Avg_Utilization_Ratio'],
      dtype='object')
```

```
test_df.shape
```

```
(2025, 21)
```

```
train_df.shape
```

```
(8102, 21)
```

```
train_df.head(5)
```

	CLIENTNUM	Attrition_Flag	Customer_Age	Gender	Dependent_count	Education_Level	Ma
0	805546308	Existing Customer	65	F	0	High School	
1	717656433	Existing Customer	50	F	2	Graduate	
2	713274183	Existing Customer	45	F	1	Graduate	
3	778819083	Attrited Customer	41	F	2	Doctorate	
4	716399583	Existing Customer	34	M	1	Uneducated	

5 rows × 21 columns

One-Hot Encoding(Needed at a later time):

Attrition_Flag, Gender, Dependent_count, Education_Level, Marital_Status, Income_Category, Card_Category

Data Cleaning

```
train_df = train_df.dropna()
test_df = test_df.dropna()
test_df.shape
```

```
(2025, 21)
```

```
train_df.shape
```

```
(8102, 21)
```

```
train_df.set_index('CLIENTNUM',inplace=True)
```

```
test_df.set_index('CLIENTNUM',inplace=True)
```

```
train_df.head(5)
```

CLIENTNUM	Attrition_Flag	Customer_Age	Gender	Dependent_count	Education_Level	Marit
805546308	Existing Customer	65	F	0	High School	
717656433	Existing Customer	50	F	2	Graduate	
713274183	Existing Customer	45	F	1	Graduate	
778819083	Attrited Customer	41	F	2	Doctorate	
716399583	Existing Customer	34	M	1	Uneducated	

Full Data Analysis

-Bar Graphs

-Mean Calculations for quantitative variables

```
full_df.columns
```

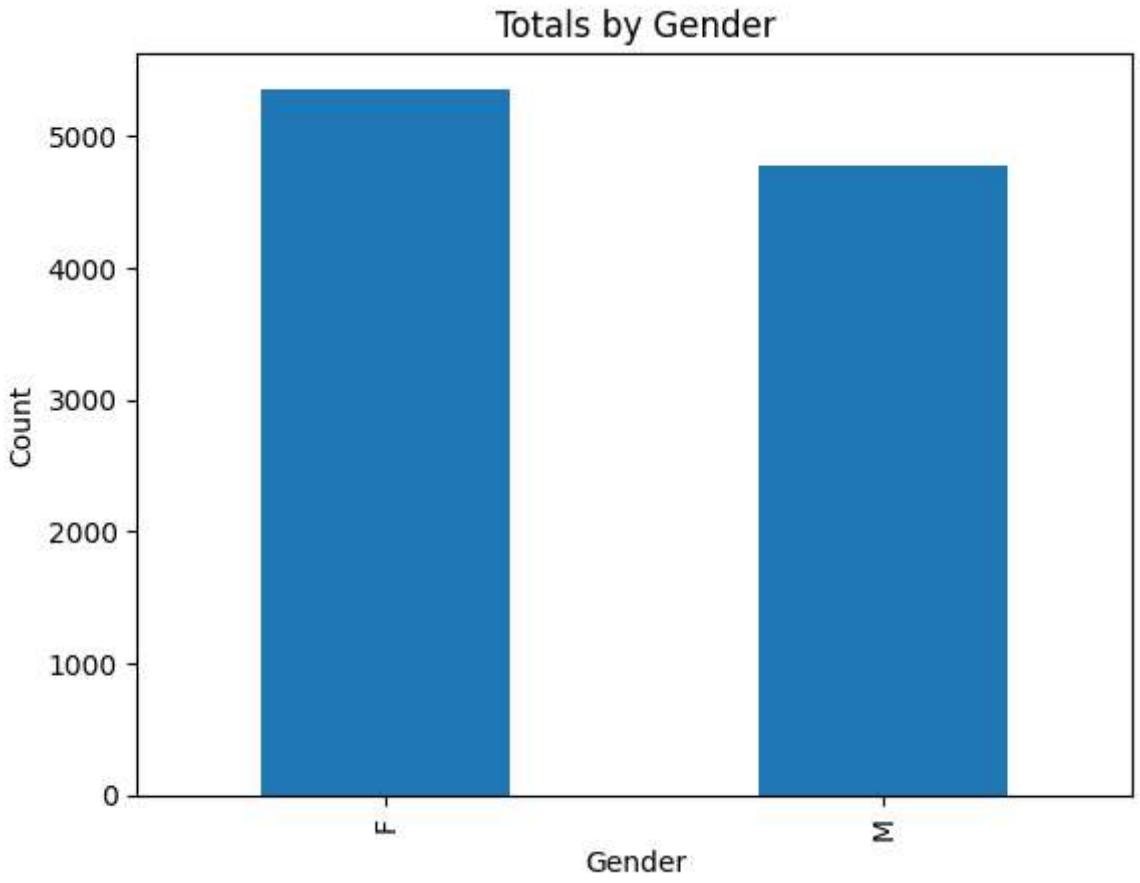
```
Index(['CLIENTNUM', 'Attrition_Flag', 'Customer_Age', 'Gender',
       'Dependent_count', 'Education_Level', 'Marital_Status',
```

```
        'Income_Category', 'Card_Category', 'Months_on_book',
        'Total_Relationship_Count', 'Months_Inactive_12_mon',
        'Contacts_Count_12_mon', 'Credit_Limit', 'Total_Revolving_Bal',
        'Avg_Open_To_Buy', 'Total_Amt_Chng_Q4_Q1', 'Total_Trans_Amt',
        'Total_Trans_Ct', 'Total_Ct_Chng_Q4_Q1', 'Avg_Utilization_Ratio',

'Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Dependent_cou

'Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Dependent_cou
      dtype='object')
```

Single Variable Analysis

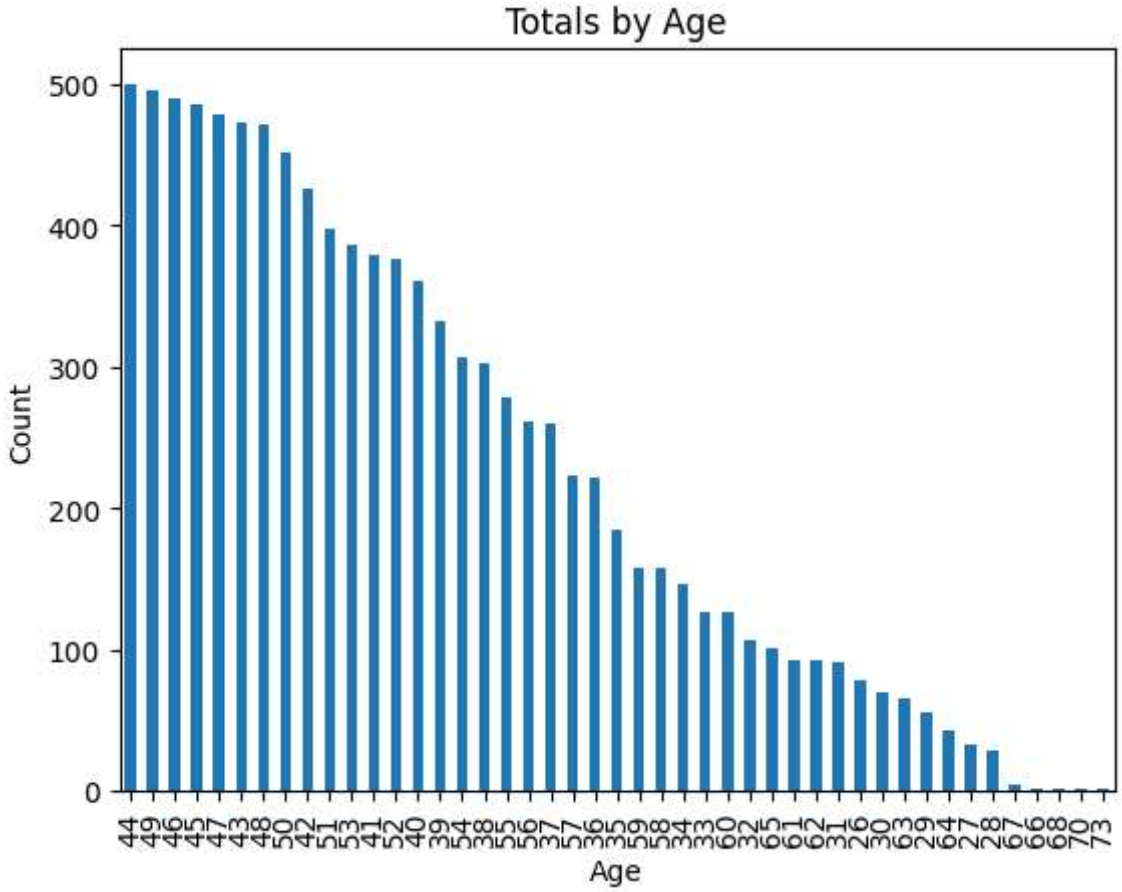
```
full_df['Gender'].value_counts().plot(kind='bar')
plt.xlabel("Gender")
plt.ylabel("Count")
plt.title("Totals by Gender")
```

```
Text(0.5, 1.0, 'Totals by Gender')
```



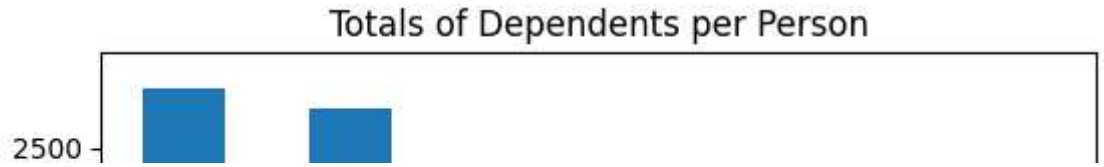

```
full_df['Customer_Age'].value_counts().plot(kind="bar")
plt.xlabel("Age")
plt.ylabel("Count")
plt.title("Totals by Age")
```

```
Text(0.5, 1.0, 'Totals by Age')
```



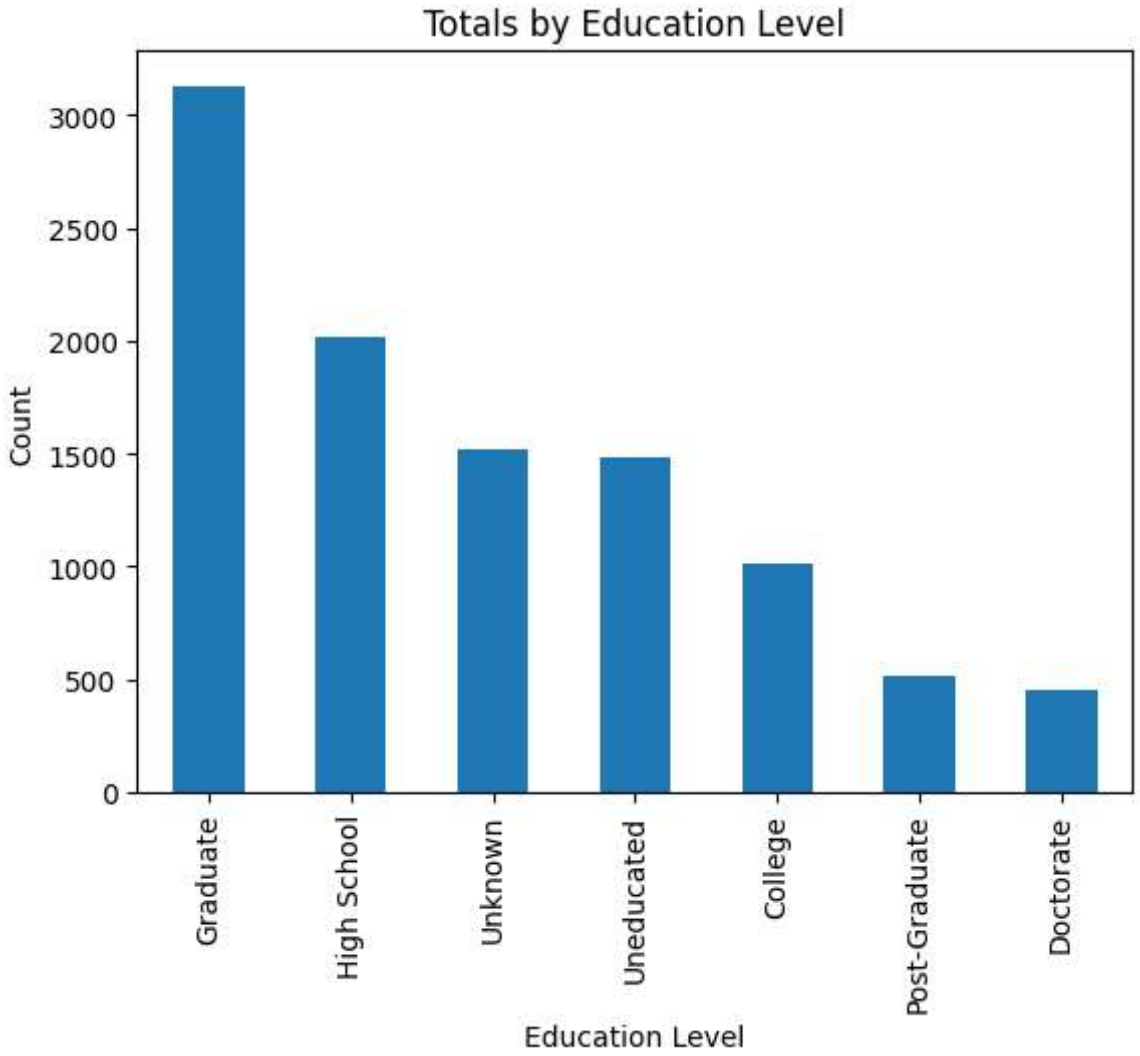

```
full_df['Dependent_count'].value_counts().plot(kind="bar")
plt.xlabel("Dependents per Person")
plt.ylabel("Count")
plt.title("Totals of Dependents per Person")
```

```
Text(0.5, 1.0, 'Totals of Dependents per Person')
```



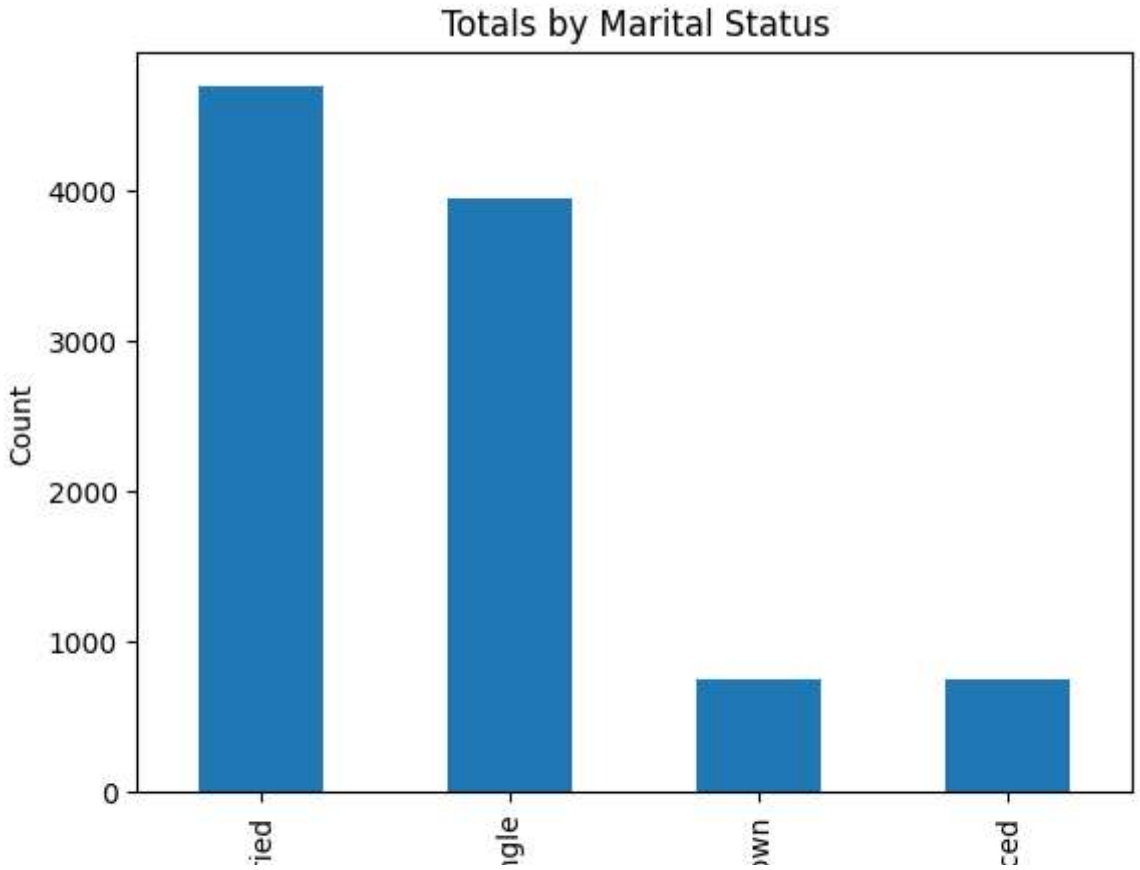

```
full_df['Education_Level'].value_counts().plot(kind="bar")
plt.xlabel("Education Level")
plt.ylabel("Count")
plt.title("Totals by Education Level")
```

```
Text(0.5, 1.0, 'Totals by Education Level')
```



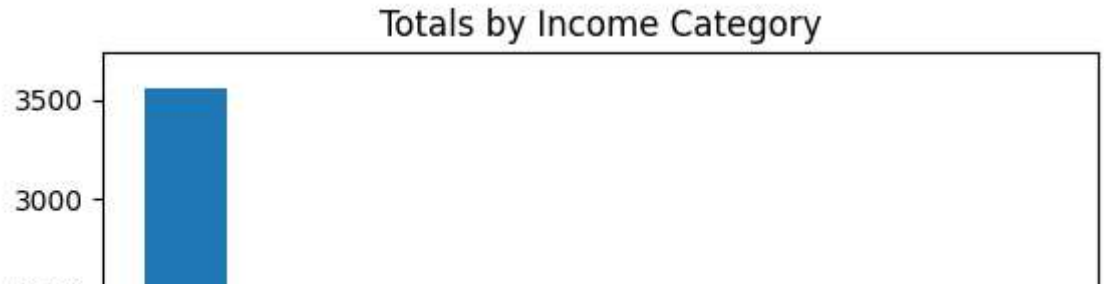

```
full_df['Marital_Status'].value_counts().plot(kind="bar")
plt.xlabel("Marital Status")
plt.ylabel("Count")
plt.title("Totals by Marital Status")
```

```
Text(0.5, 1.0, 'Totals by Marital Status')
```



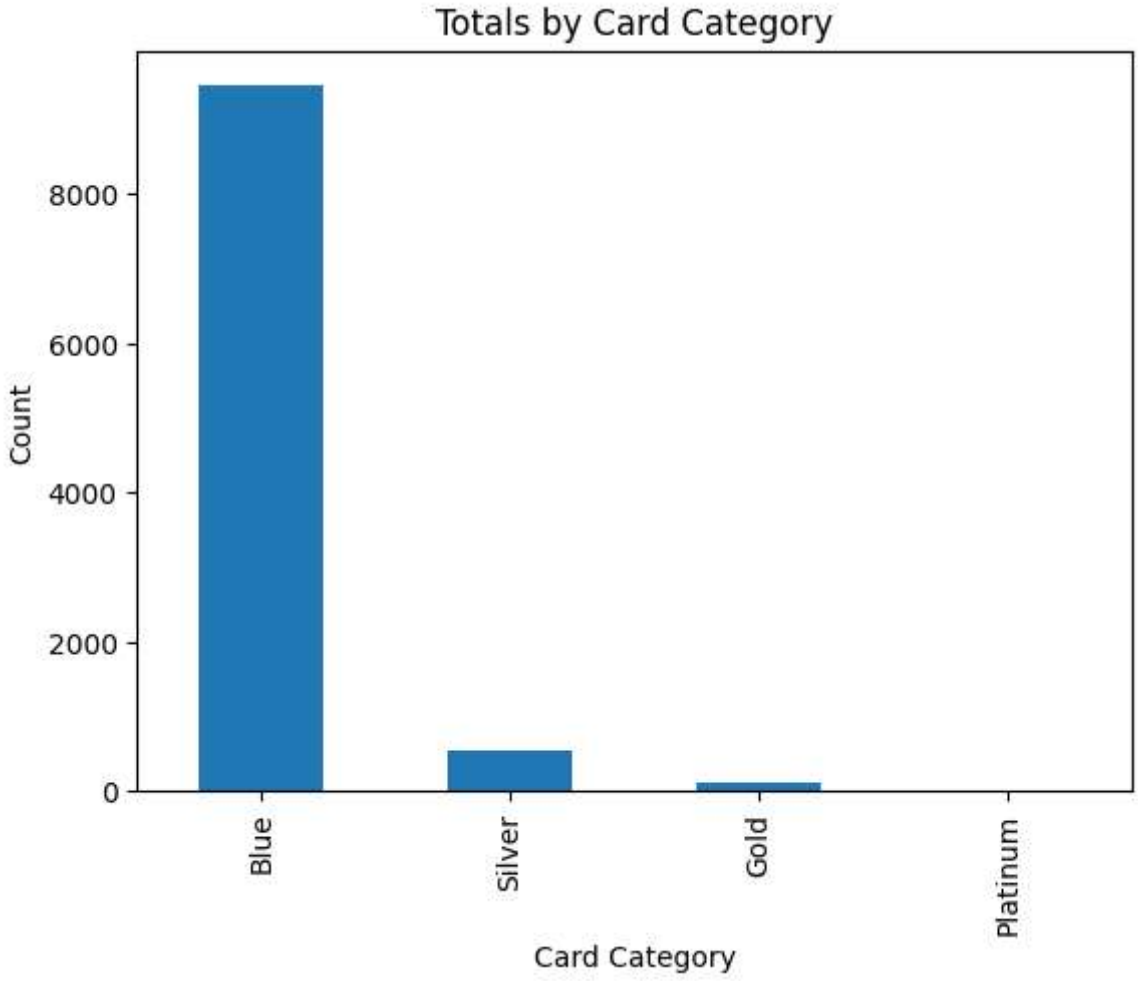

```
full_df['Income_Category'].value_counts().plot(kind="bar")
plt.xlabel("Income Category")
plt.ylabel("Count")
plt.title("Totals by Income Category")
```

```
Text(0.5, 1.0, 'Totals by Income Category')
```




```
full_df['Card_Category'].value_counts().plot(kind="bar")
plt.xlabel("Card Category")
plt.ylabel("Count")
plt.title("Totals by Card Category")
```

```
Text(0.5, 1.0, 'Totals by Card Category')
```



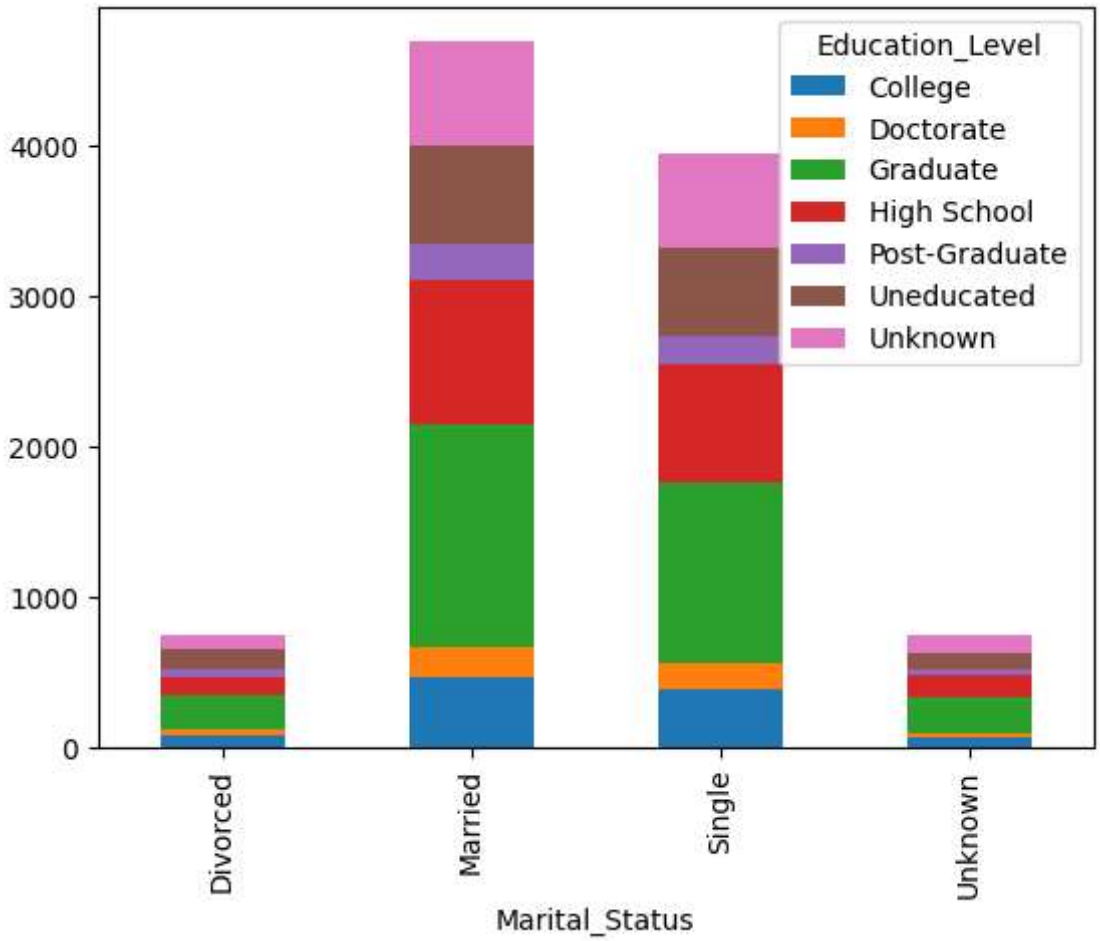
Credit Card Churn Drivers- Multivariable Analysis

```
agg_gender = full_df.groupby(['Marital_Status','Education_Level'])['Gender'].count().unstack(
agg_gender.head(5)
```

Education_Level	College	Doctorate	Graduate	High School	Post-Graduate	Uneducated	Unknown
Marital_Status							
Divorced	86	36	225	128	41	136	96
Married	467	205	1479	949	243	656	688
Single	386	182	1197	782	189	586	621

```
agg_gender.plot(kind="bar",stacked=True)
```

```
<Axes: xlabel='Marital_Status'>
```

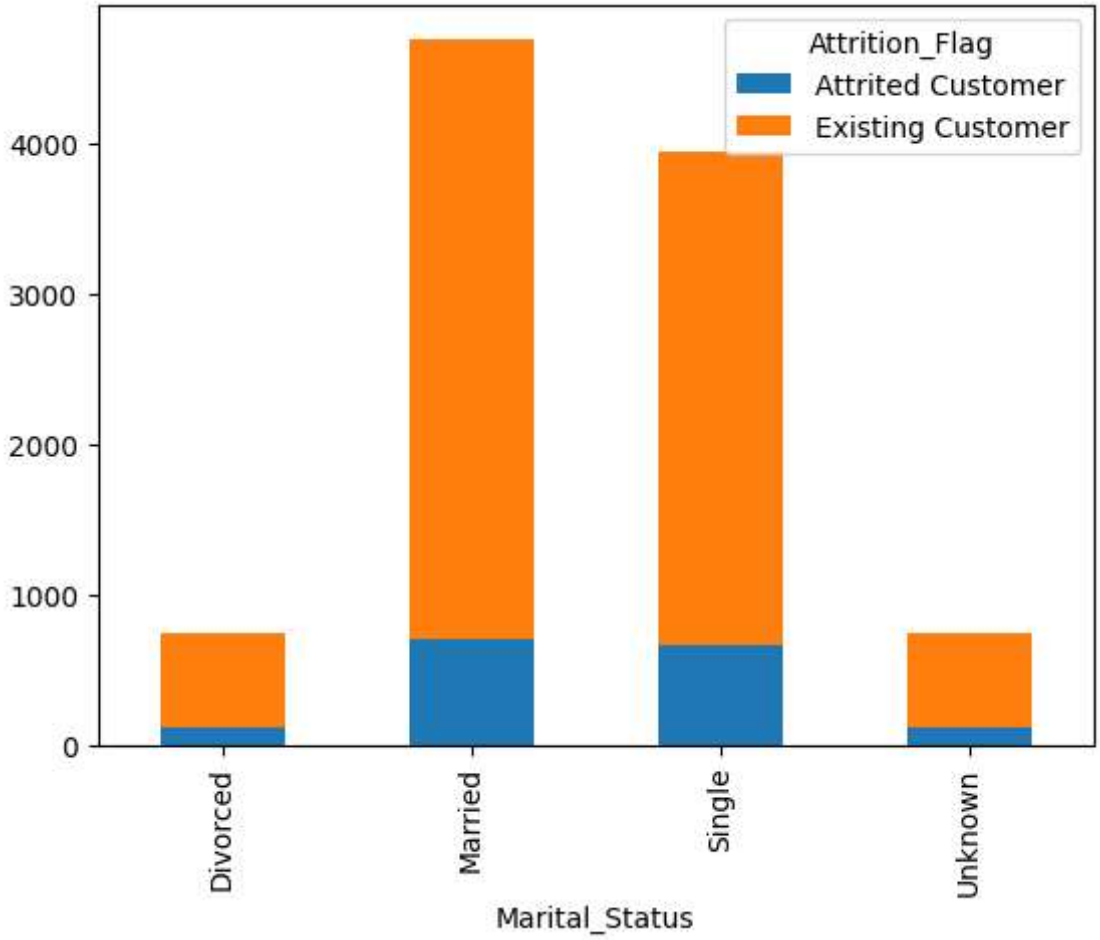



```
agg_customers = full_df.groupby(['Marital_Status','Attrition_Flag'])['Attrition_Flag'].count(
agg_customers.drop("Unknown",axis=0)
agg_customers.head(5)
```

Attrition_Flag	Attrited Customer	Existing Customer
Marital_Status		

```
agg_customers.plot(kind="bar",stacked=True)
```

```
<Axes: xlabel='Marital_Status'>
```

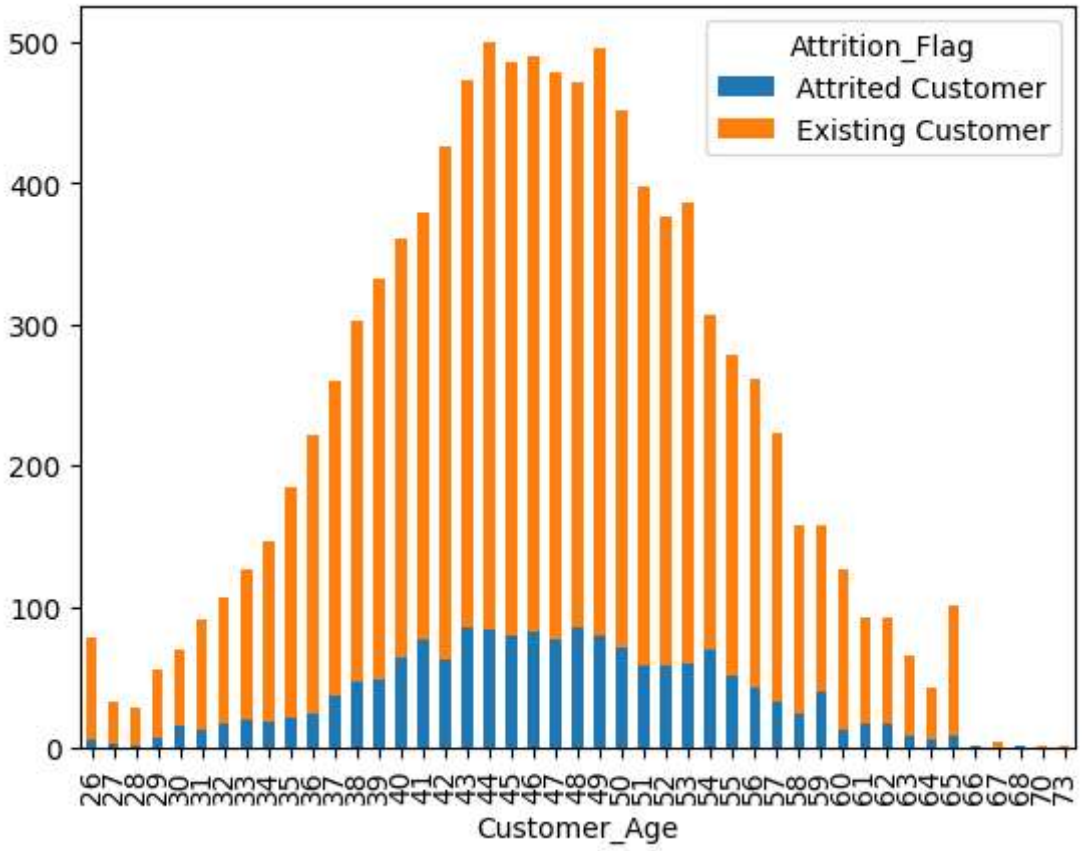



```
agg_customers2 = full_df.groupby(['Customer_Age','Attrition_Flag'])['Attrition_Flag'].count()
agg_customers2.head(5)
```

Attrition_Flag	Attrited Customer	Existing Customer
Customer_Age		
26	6.0	72.0
27	3.0	29.0
28	1.0	28.0
29	7.0	49.0
30	15.0	55.0

```
agg_customers2.plot(kind="bar",stacked=True)
```

```
<Axes: xlabel='Customer_Age'>
```



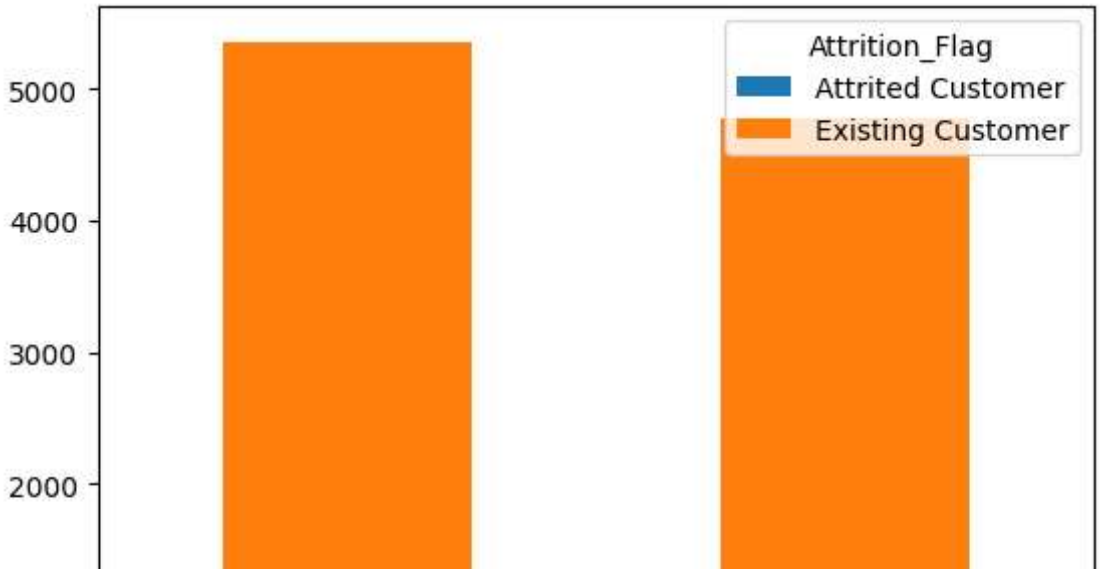
The graph for age shows nearly a perfect bell curve without any normalization, this may indicate that age is a contributing factor to credit card turnover

```
agg_customers3 = full_df.groupby(['Gender','Attrition_Flag'])['Attrition_Flag'].count().unsta
agg_customers3.head(5)
```

Attrition_Flag	Attrited Customer	Existing Customer
Gender		
F	930	4428
M	697	4072

```
agg_customers3.plot(kind="bar",stacked=True)
```

```
<Axes: xlabel='Gender'>
```




```
agg_customers4 = full_df.groupby(['Dependent_count','Attrition_Flag'])['Attrition_Flag'].coun
agg_customers4.head(5)
```

Attrition_Flag	Attrited Customer	Existing Customer
Dependent_count		
0	135	769
1	269	1569
2	417	2238
3	482	2250
4	260	1314

```
agg_customers4.plot(kind="bar",stacked=True)
```

```
<Axes: xlabel='Dependent_count'>
```


```
agg_customers5 = full_df.groupby(['Education_Level','Attrition_Flag'])['Attrition_Flag'].coun
agg_customers5.head(5)
```

Attrition_Flag	Attrited Customer	Existing Customer
Education_Level		
College	154	859
Doctorate	95	356
Graduate	487	2641
High School	306	1707
Post-Graduate	92	424

```
agg_customers5.plot(kind="bar",stacked=True)
```

```
<Axes: xlabel='Education_Level'>
```


```
agg_customers6 = full_df.groupby(['Income_Category','Attrition_Flag'])['Attrition_Flag'].coun
agg_customers6.head(5)
```

Attrition_Flag	Attrited Customer	Existing Customer
Income_Category		
$120K +	126	601
$40K - $60K	271	1519
$60K - $80K	189	1213
$80K - $120K	242	1293
Less than $40K	612	2949

```
agg_customers6.plot(kind="bar",stacked=True)
```

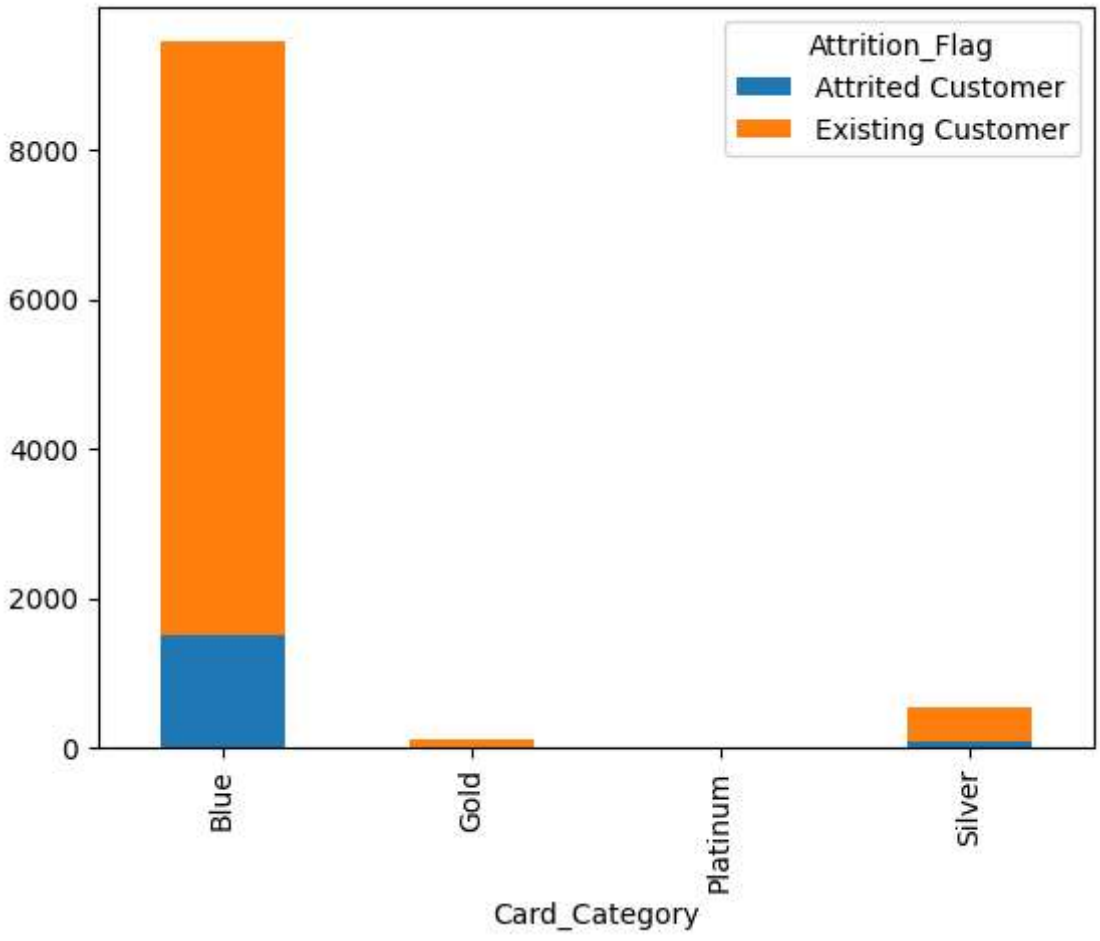
```
<Axes: xlabel='Income_Category'>
```

```
agg_customers7 = full_df.groupby(['Card_Category','Attrition_Flag'])['Attrition_Flag'].count(
agg_customers7.head(5)
```

Attrition_Flag	Attrited Customer	Existing Customer
Card_Category		
Blue	1519	7917
Gold	21	95
Platinum	5	15
Silver	82	473

```
agg_customers7.plot(kind="bar",stacked=True)
```

```
<Axes: xlabel='Card_Category'>
```

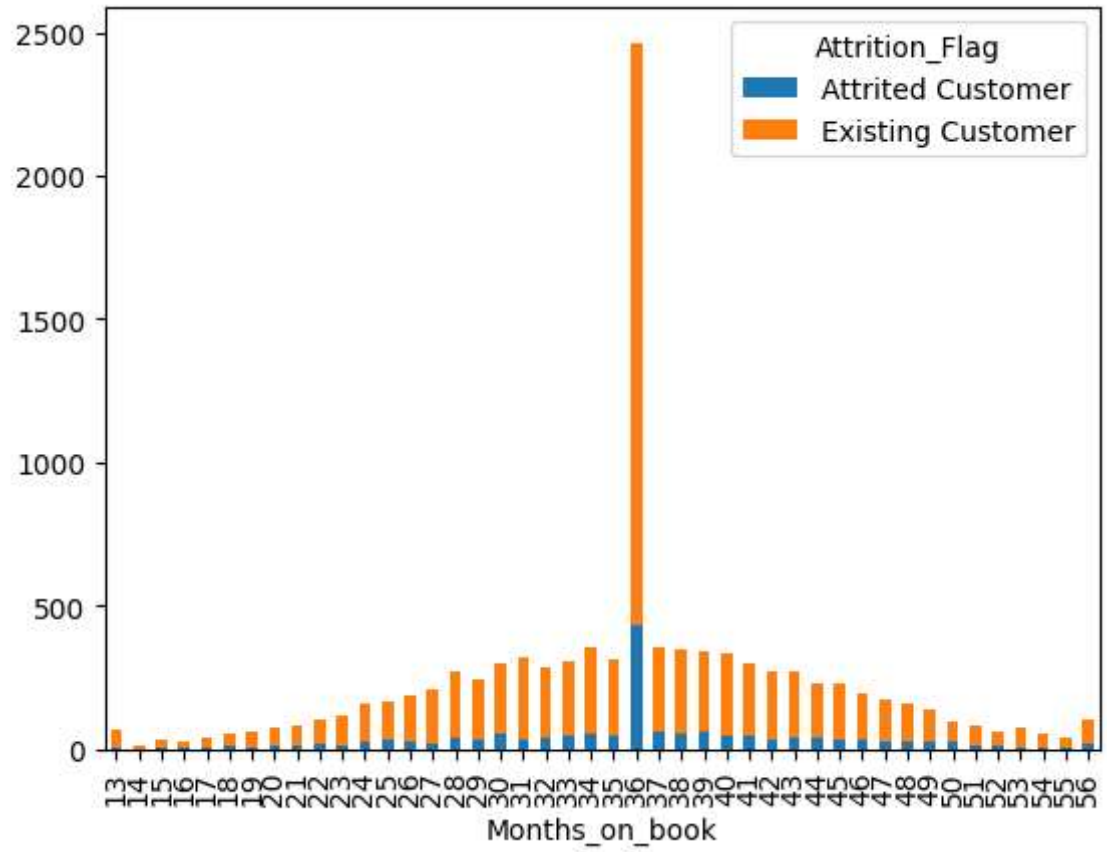



```
agg_customers8 = full_df.groupby(['Months_on_book','Attrition_Flag'])['Attrition_Flag'].count
agg_customers8
```

Attrition_Flag Months_on_book	Attrited Customer	Existing Customer
13	7	63
14	1	15
15	9	25
16	3	26
17	4	35
18	13	45
19	6	57
20	13	61
21	10	73
22	20	85
23	12	104
24	28	132
25	31	134
26	24	162
27	23	183
28	43	232
29	34	207
30	58	242
31	34	284
32	44	245
33	48	257
34	57	296
35	45	272
36	430	2033
37	62	296
38	57	290
39	64	277
40	45	288
41	51	246

```
agg_customers8.plot(kind="bar",stacked=True)
```

```
<Axes: xlabel='Months_on_book'>
```

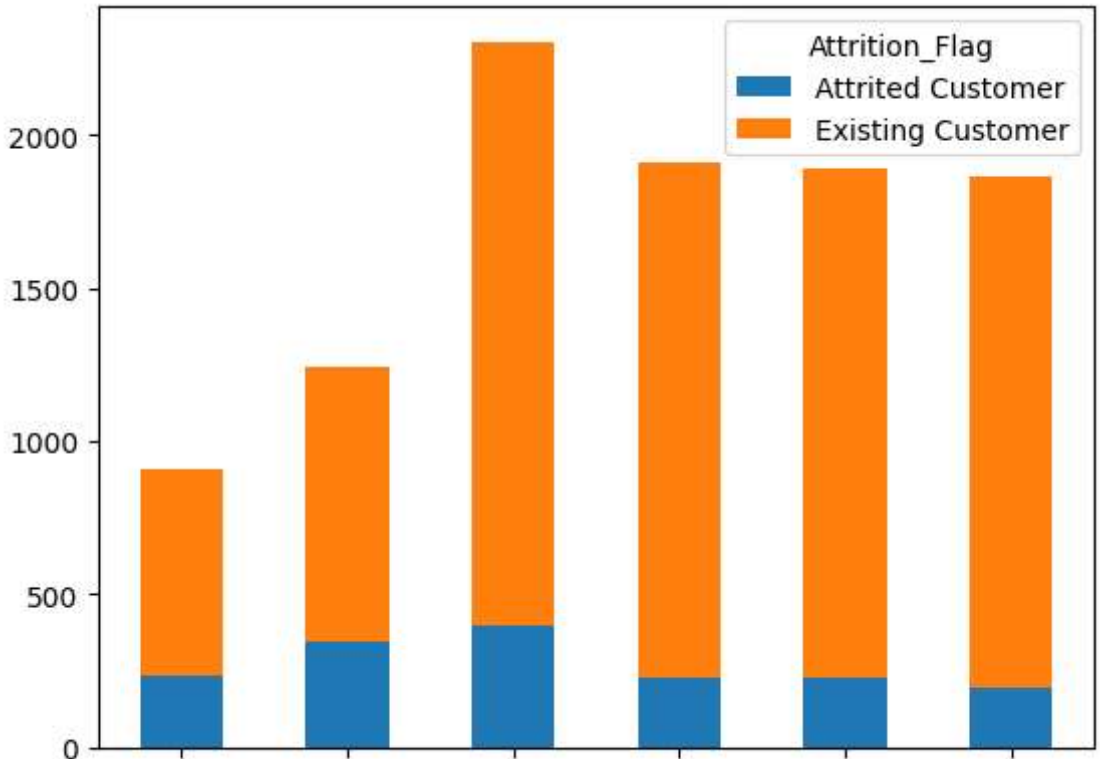



```
agg_customers9 = full_df.groupby(['Total_Relationship_Count','Attrition_Flag'])['Attrition_Fl
agg_customers9
```

Attrition_Flag	Attrited Customer	Existing Customer
Total_Relationship_Count		
1	233	677
2	346	897
3	400	1905
4	225	1687
5	227	1664
6	196	1670

```
agg_customers9.plot(kind="bar",stacked=True)
```

```
<Axes: xlabel='Total_Relationship_Count'>
```



Year 3(36 Month) Analysis

There are signifigantly higher values of exisitng and attritied customers at the 3 year mark than in any other month_on_book value. We will explore if the trend from above true or if they based based on the high customer volume

```
year_3_df = full_df[full_df['Months_on_book'] == 36]
year_3_df.shape
```

```
(2463, 23)
```

```
agg_customers8_1 = year_3_df.groupby(['Months_on_book','Attrition_Flag'])['Attrition_Flag'].c
agg_customers8_1
```

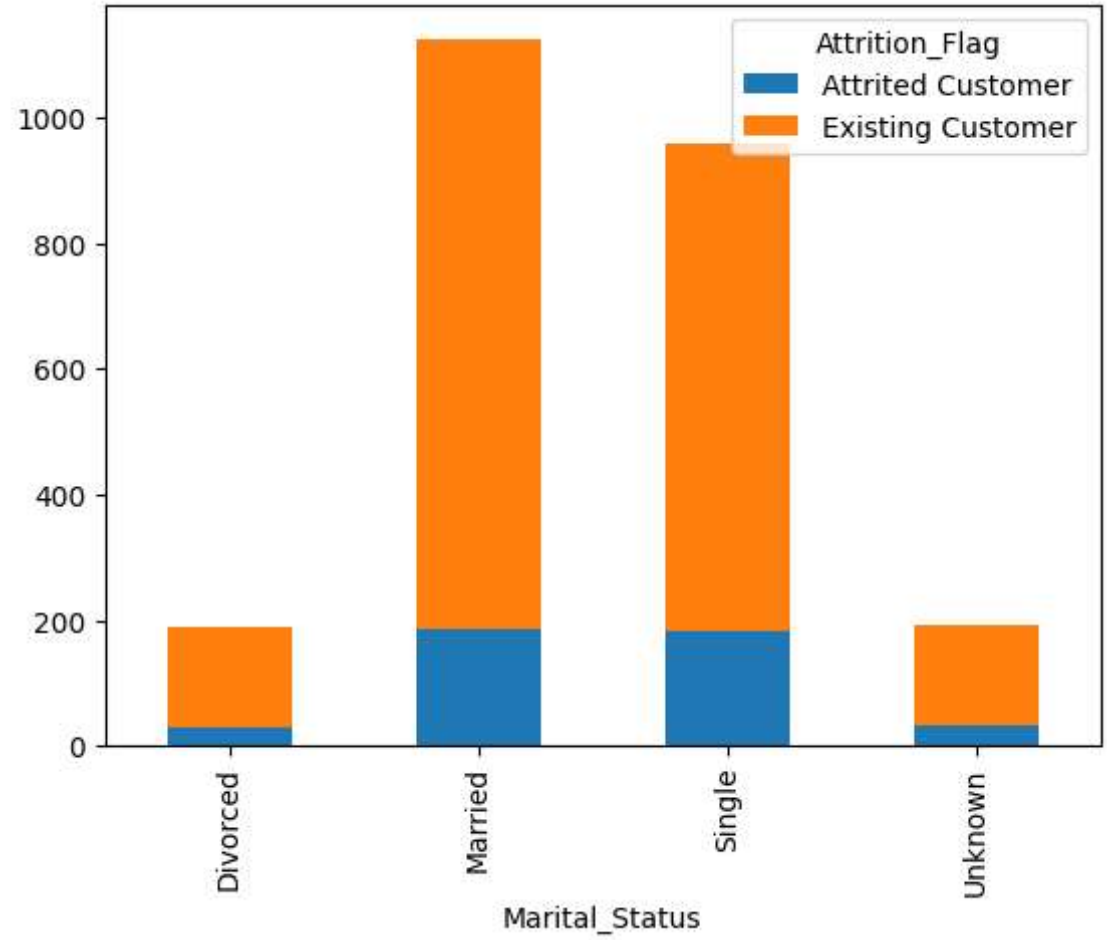
Attrition_Flag	Attrited Customer	Existing Customer
Months_on_book		
36	430	2033

```
agg3_customers_1 = year_3_df.groupby(['Marital_Status','Attrition_Flag'])['Attrition_Flag'].c
agg3_customers_1.drop("Unknown",axis=0)
agg3_customers_1.head(5)
```

Attrition_Flag	Attrited Customer	Existing Customer
Marital_Status		
Divorced	29	159
Married	185	940
Single	184	774

```
agg3_customers_1.plot(kind="bar",stacked=True)
```

```
<Axes: xlabel='Marital_Status'>
```

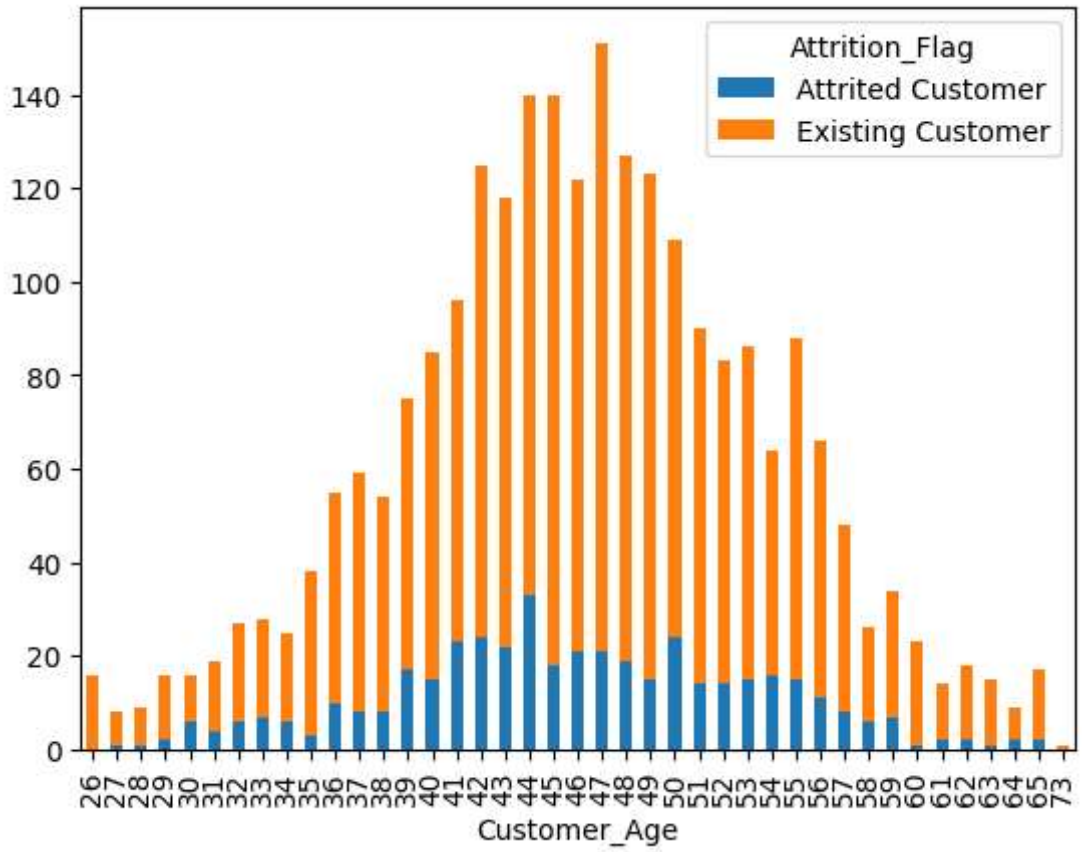



```
agg3_customers_2 = year_3_df.groupby(['Customer_Age','Attrition_Flag'])['Attrition_Flag'].cou
agg3_customers_2.head(5)
```

Attrition_Flag	Attrited Customer	Existing Customer

```
agg3_customers_2.plot(kind="bar",stacked=True)
```

```
<Axes: xlabel='Customer_Age'>
```

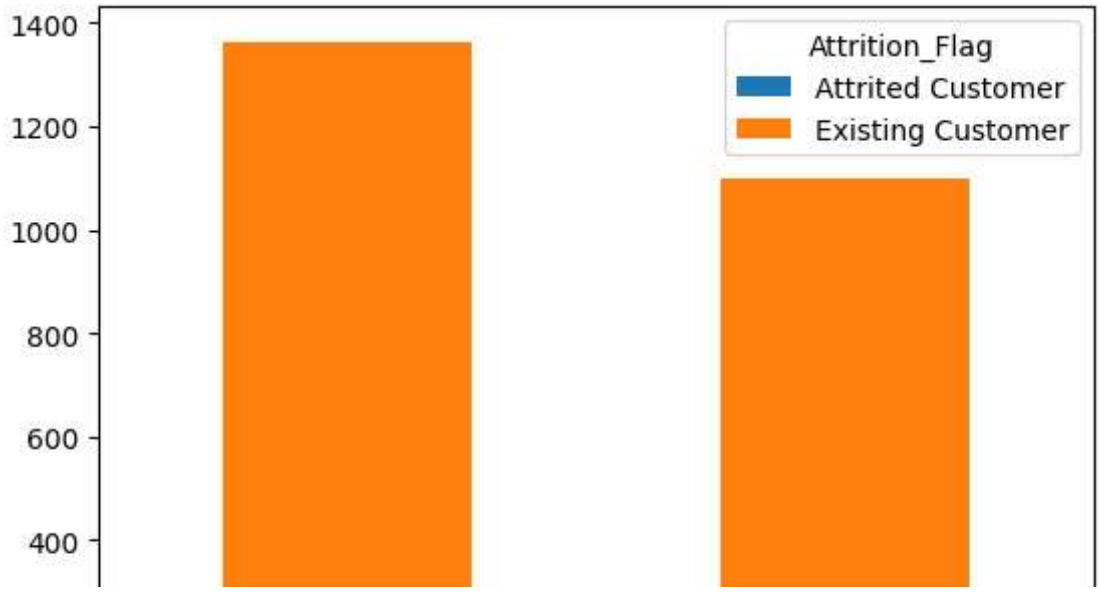



```
agg3_customers_3 = year_3_df.groupby(['Gender','Attrition_Flag'])['Attrition_Flag'].count().u
agg3_customers_3.head(5)
```

Attrition_Flag	Attrited Customer	Existing Customer
Gender		
F	260	1104
M	170	929

```
agg3_customers_3.plot(kind="bar",stacked=True)
```

```
<Axes: xlabel='Gender'>
```




```
agg3_customers_4 = year_3_df.groupby(['Dependent_count','Attrition_Flag'])['Attrition_Flag'].
agg3_customers_4.head(5)
```

Attrition_Flag	Attrited Customer	Existing Customer
Dependent_count		
0	23	194
1	78	338
2	106	506
3	136	555
4	71	354

```
agg3_customers_4.plot(kind="bar",stacked=True)
```

```
<Axes: xlabel='Dependent_count'>
```


```
agg3_customers_5 = year_3_df.groupby(['Education_Level','Attrition_Flag'])['Attrition_Flag'].
agg3_customers_5.head(5)
```

Attrition_Flag	Attrited Customer	Existing Customer
Education_Level		
College	37	188
Doctorate	21	84
Graduate	134	595
High School	78	431
Post-Graduate	25	103

```
agg3_customers_5.plot(kind="bar",stacked=True)
```

```
<Axes: xlabel='Education_Level'>
```


```
agg3_customers_6 = year_3_df.groupby(['Income_Category','Attrition_Flag'])['Attrition_Flag'].
agg3_customers_6.head(5)
```

Attrition_Flag	Attrited Customer	Existing Customer
Income_Category		
$120K +	29	127
$40K - $60K	56	340
$60K - $80K	46	295
$80K - $120K	61	289
Less than $40K	191	753

```
agg3_customers_6.plot(kind="bar",stacked=True)
```

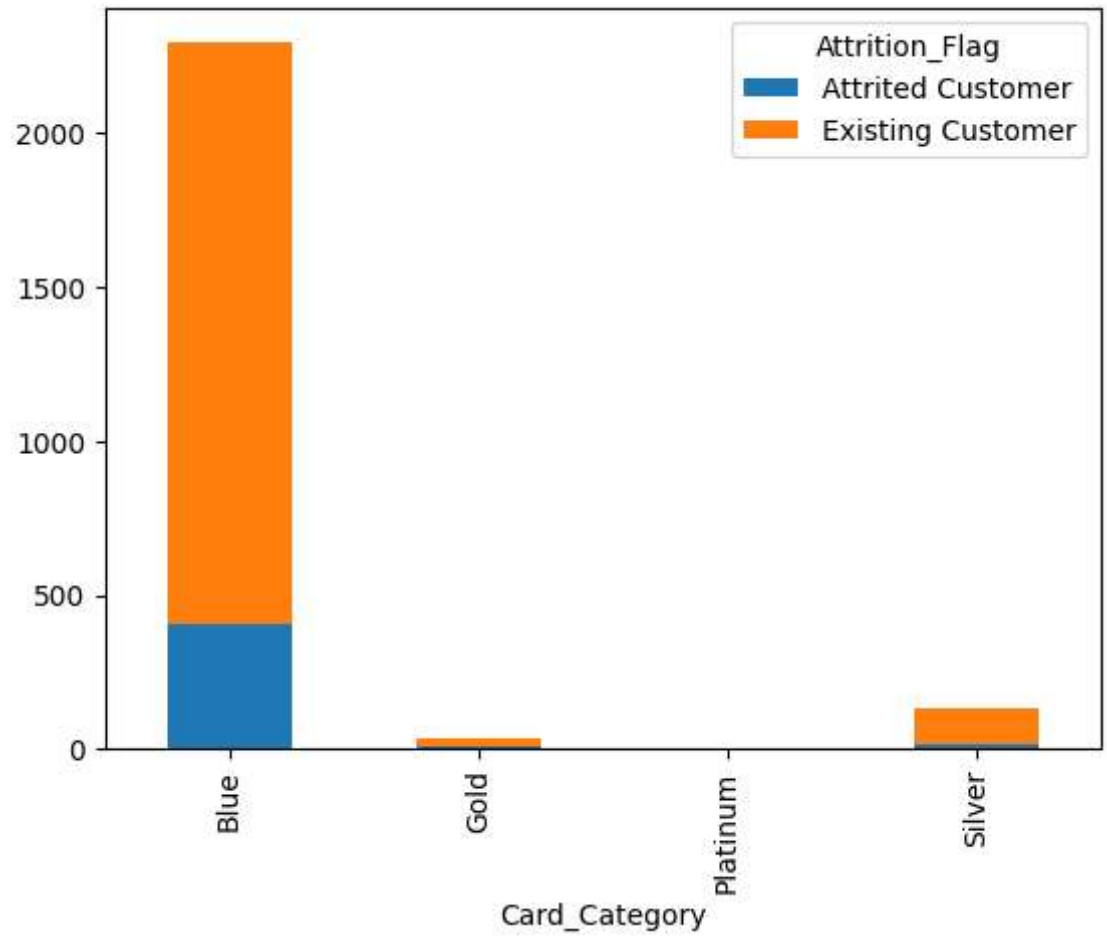
```
<Axes: xlabel='Income_Category'>
```

```
agg3_customers_7 = year_3_df.groupby(['Card_Category','Attrition_Flag'])['Attrition_Flag'].co
agg3_customers_7.head(5)
```

Attrition_Flag	Attrited Customer	Existing Customer
Card_Category		
Blue	405.0	1888.0
Gold	8.0	27.0
Platinum	NaN	2.0
Silver	17.0	116.0

```
agg3_customers_7.plot(kind="bar",stacked=True)
```

```
<Axes: xlabel='Card_Category'>
```

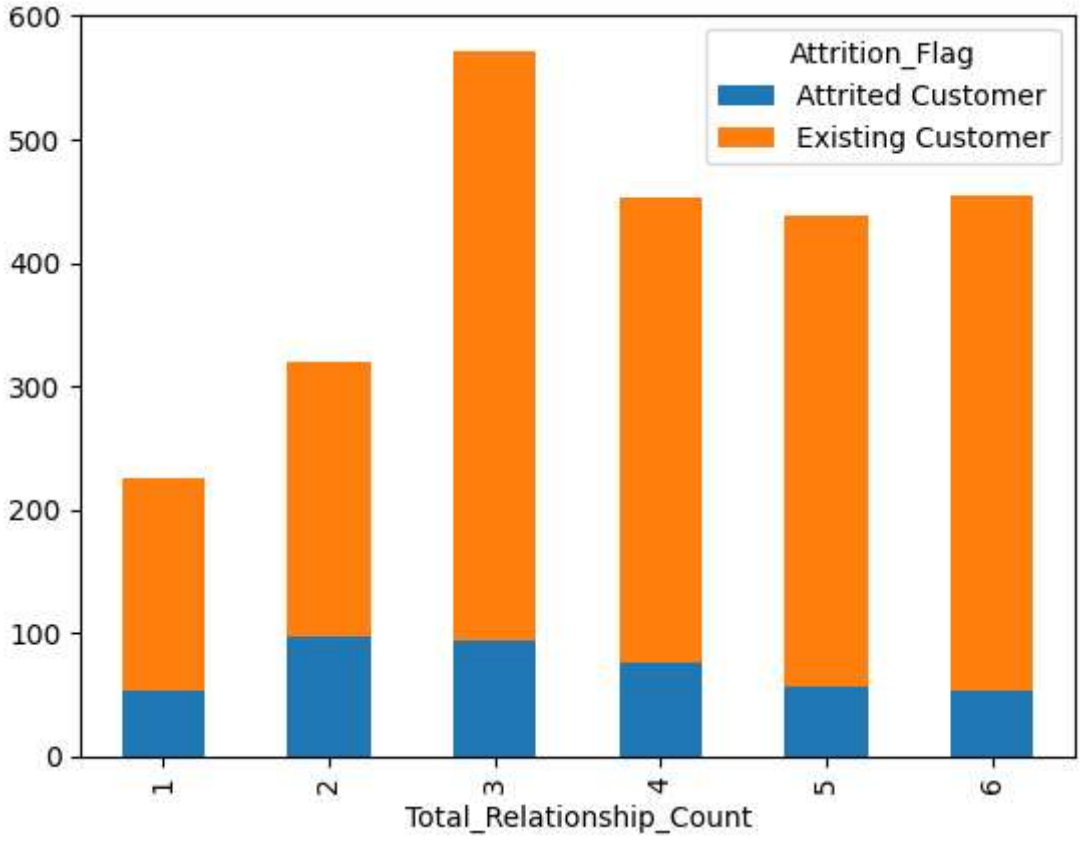



```
agg3_customers_8 = year_3_df.groupby(['Total_Relationship_Count','Attrition_Flag'])['Attritio
agg3_customers_8.head(5)
```

Attrition_Flag	Attrited Customer	Existing Customer
Total_Relationship_Count		
1	53	172
2	97	222
3	94	478
4	76	377

```
agg3_customers_8.plot(kind="bar",stacked=True)
```

```
<Axes: xlabel='Total_Relationship_Count'>
```



Model Building

Cassie Kamens and Matt Werner

BAN/DAT Project

Credit Card Customer Turnover

One-Hot Encoding

```
import seaborn as sns
from sklearn.preprocessing import OneHotEncoder
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.model_selection import train_test_split
import io

from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.model_selection import train_test_split, cross_val_score
```

Train df encoding

```
encode_DC=OneHotEncoder(sparse=False)
mat_DC=encode_DC.fit_transform(train_df["Dependent_count"].to_numpy().reshape(-1,1))
df_DC=pd.DataFrame(mat_DC,columns=encode_DC.categories_)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWa
  warnings.warn(
```

```
encode_EL=OneHotEncoder(sparse=False)
mat_EL=encode_EL.fit_transform(train_df["Education_Level"].to_numpy().reshape(-1,1))
df_EL=pd.DataFrame(mat_EL,columns=encode_EL.categories_)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWa
  warnings.warn(
```

```
encode_MarS=OneHotEncoder(sparse=False)
mat_MarS=encode_MarS.fit_transform(train_df["Marital_Status"].to_numpy().reshape(-1,1))
df_MarS=pd.DataFrame(mat_MarS,columns=encode_MarS.categories_)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWa
  warnings.warn(
```

```
encode_IC=OneHotEncoder(sparse=False)
mat_IC=encode_IC.fit_transform(train_df["Income_Category"].to_numpy().reshape(-1,1))
df_IC=pd.DataFrame(mat_IC,columns=encode_IC.categories_)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWa
  warnings.warn(
```

```
encode_CC=OneHotEncoder(sparse=False)
mat_CC=encode_CC.fit_transform(train_df["Card_Category"].to_numpy().reshape(-1,1))
df_CC=pd.DataFrame(mat_CC,columns=encode_CC.categories_)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWa
  warnings.warn(
```

```
encode_TRC=OneHotEncoder(sparse=False)
mat_TRC=encode_TRC.fit_transform(train_df["Total_Relationship_Count"].to_numpy().reshape(-1,1
df_TRC=pd.DataFrame(mat_TRC,columns=encode_TRC.categories_)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWa
  warnings.warn(
```

```
df_cats=pd.concat([df_DC,df_EL,df_MarS,df_IC,df_CC,df_TRC], axis=1)
df_cats
```

	0	1	2	3	4	5	College	Doctorate	Graduate	High School	...	Blue	Gold	P
0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	...	1.0	0.0	
1	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	1.0	0.0	
2	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	1.0	0.0	
3	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	...	1.0	0.0	
4	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	1.0	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
8097	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	
8098	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	1.0	0.0	
8099	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	1.0	0.0	
8100	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	1.0	0.0	
8101	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	...	1.0	0.0	

Train df Standard Scaling

```
num_pipeline = Pipeline([
        ('imputer', SimpleImputer(strategy="median")),
        ('std_scaler', StandardScaler()),
    ])
```

```
credit_train_continuous= num_pipeline.fit_transform(train_df[['Months_on_book','Months_Inacti
credit_train_continuous=pd.DataFrame(credit_train_continuous,columns=['Months_on_book','Month
```

Combining Train DF

```
credit_train_final=pd.concat([credit_train_continuous,df_cats],axis=1)
```

Test Encoding

```
encode_DC=OneHotEncoder(sparse=False)
mat_DC=encode_DC.fit_transform(test_df["Dependent_count"].to_numpy().reshape(-1,1))
df_DC1=pd.DataFrame(mat_DC,columns=encode_DC.categories_)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWa
  warnings.warn(
```

```
encode_EL=OneHotEncoder(sparse=False)
mat_EL=encode_EL.fit_transform(test_df["Education_Level"].to_numpy().reshape(-1,1))
df_EL1=pd.DataFrame(mat_EL,columns=encode_EL.categories_)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWa
  warnings.warn(
```

```
encode_MarS=OneHotEncoder(sparse=False)
mat_MarS=encode_MarS.fit_transform(test_df["Marital_Status"].to_numpy().reshape(-1,1))
df_MarS1=pd.DataFrame(mat_MarS,columns=encode_MarS.categories_)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWa
  warnings.warn(
```

```
encode_IC=OneHotEncoder(sparse=False)
mat_IC=encode_IC.fit_transform(test_df["Income_Category"].to_numpy().reshape(-1,1))
df_IC1=pd.DataFrame(mat_IC,columns=encode_IC.categories_)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWa
  warnings.warn(
```

```
encode_CC=OneHotEncoder(sparse=False)
mat_CC=encode_CC.fit_transform(test_df["Card_Category"].to_numpy().reshape(-1,1))
df_CC1=pd.DataFrame(mat_CC,columns=encode_CC.categories_)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWa
  warnings.warn(
```

```
encode_TRC=OneHotEncoder(sparse=False)
mat_TRC=encode_TRC.fit_transform(test_df["Total_Relationship_Count"].to_numpy().reshape(-1,1)
df_TRC1=pd.DataFrame(mat_TRC,columns=encode_TRC.categories_)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWa
  warnings.warn(
```

```
df_cats1=pd.concat([df_DC1,df_EL1,df_MarS1,df_IC1,df_CC1,df_TRC1], axis=1)
df_cats1
```

	0	1	2	3	4	5	College	Doctorate	Graduate	High School	...	Blue	Gold	P
0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	1.0	0.0	
1	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	1.0	0.0	
2	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	1.0	0.0	
3	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	...	1.0	0.0	
4	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	1.0	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
2020	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	...	1.0	0.0	
2021	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	...	1.0	0.0	
2022	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	1.0	0.0	
2023	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	1.0	0.0	
2024	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	1.0	0.0	

Test Standard Scale

```
num_pipeline = Pipeline([
        ('imputer', SimpleImputer(strategy="median")),
        ('std_scaler', StandardScaler()),
    ])
```

```
credit_test_continuous= num_pipeline.fit_transform(test_df[['Months_on_book','Months_Inactive
credit_test_continuous=pd.DataFrame(credit_train_continuous,columns=['Months_on_book','Months
```

Combine Test DF

```
credit_test_final=pd.concat([credit_test_continuous,df_cats1],axis=1)
```

Dataframe Name Assignment

```
y_train = train_df['Attrition_Flag']
```

```
x_train = credit_train_final
```

```
y_test = credit_test_final
```

Nueral Net Model

```
x_train_2, x_valid, y_train_2, y_valid = train_test_split(x_train, y_train, test_size=0.2, ra
```

```
from sklearn.neural_network import MLPClassifier
clf = MLPClassifier(solver='adam', alpha=1e-5, random_state=1, max_iter=500,hidden_layer_size

clf.fit(x_train.values, y_train)
```

```
Iteration 182, loss = 0.10647989
Iteration 183, loss = 0.10610979
Iteration 184, loss = 0.10642233
Iteration 185, loss = 0.10633652
Iteration 186, loss = 0.10596419
Iteration 187, loss = 0.10550447
Iteration 188, loss = 0.10649064
Iteration 189, loss = 0.10623044
Iteration 190, loss = 0.10676218
Iteration 191, loss = 0.10573563
Iteration 192, loss = 0.10527745
Iteration 193, loss = 0.10551886
Iteration 194, loss = 0.10529568
Iteration 195, loss = 0.10528037
Iteration 196, loss = 0.10714332
Iteration 197, loss = 0.10468961
Iteration 198, loss = 0.10441395
Iteration 199, loss = 0.10535654
Iteration 200, loss = 0.10513559
Iteration 201, loss = 0.10427813
Iteration 202, loss = 0.10444775
Iteration 203, loss = 0.10437686
Iteration 204, loss = 0.10447823
Iteration 205, loss = 0.10413197
Iteration 206, loss = 0.10431540
Iteration 207, loss = 0.10415417
Iteration 208, loss = 0.10507567
Iteration 209, loss = 0.10388173
Iteration 210, loss = 0.10387955
Iteration 211, loss = 0.10371302
Iteration 212, loss = 0.10415990
Iteration 213, loss = 0.10329010
Iteration 214, loss = 0.10443512
Iteration 215, loss = 0.10318052
Iteration 216, loss = 0.10422668
Iteration 217, loss = 0.10373798
Iteration 218, loss = 0.10264110
Iteration 219, loss = 0.10345543
Iteration 220, loss = 0.10269512
Iteration 221, loss = 0.10347428
Iteration 222, loss = 0.10211709
Iteration 223, loss = 0.10276862
Iteration 224, loss = 0.10290627
Iteration 225, loss = 0.10253343
Iteration 226, loss = 0.10297115
Iteration 227, loss = 0.10228074
Iteration 228, loss = 0.10298965
Iteration 229, loss = 0.10242505
Iteration 230, loss = 0.10234880
Iteration 231, loss = 0.10222775
Iteration 232, loss = 0.10280726
Iteration 233, loss = 0.10224993
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopp
```

MLPClassifier

```
MLPClassifier(alpha=1e-05, hidden_layer_sizes=(13, 6), max_iter=500,
              random_state=1, verbose=True)
```

Model Preformance

```
clf1 = clf.fit(x_train_2.values, y_train_2)
y_pred = clf1.predict(x_valid.values)

print (accuracy_score(y_valid,y_pred))

neur_acc = accuracy_score(y_valid,y_pred)
```

```
Iteration 404, loss = 0.07138789
Iteration 405, loss = 0.07237895
Iteration 406, loss = 0.07252019
Iteration 407, loss = 0.07192169
Iteration 408, loss = 0.07164275
Iteration 409, loss = 0.07119523
Iteration 410, loss = 0.07161371
Iteration 411, loss = 0.07186499
Iteration 412, loss = 0.07111977
Iteration 413, loss = 0.07217954
Iteration 414, loss = 0.07100689
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stop
```

Permutation Importance

```
!pip install eli5
!pip install shap
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/pub
Collecting eli5
  Downloading eli5-0.13.0.tar.gz (216 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 216.2/216.2 kB 10.8 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: attrs>17.1.0 in /usr/local/lib/python3.10/dist-packages (
Requirement already satisfied: jinja2>=3.0.0 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: numpy>=1.9.0 in /usr/local/lib/python3.10/dist-packages (
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from el
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from eli5
Requirement already satisfied: scikit-learn>=0.20 in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: graphviz in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: tabulate>=0.7.7 in /usr/local/lib/python3.10/dist-package
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-package
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-pa
Building wheels for collected packages: eli5
  Building wheel for eli5 (setup.py) ... done
  Created wheel for eli5: filename=eli5-0.13.0-py2.py3-none-any.whl size=107747 sha256=8
  Stored in directory: /root/.cache/pip/wheels/b8/58/ef/2cf4c306898c2338d51540e0922c8e0c
Successfully built eli5
Installing collected packages: eli5
Successfully installed eli5-0.13.0
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/pub
Collecting shap
  Downloading shap-0.41.0-cp310-cp310-manylinux_2_12_x86_64.manylinux2010_x86_64.whl (57
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 572.6/572.6 kB 22.5 MB/s eta 0:00:00
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (
Requirement already satisfied: numba in /usr/local/lib/python3.10/dist-packages (from sh
Requirement already satisfied: packaging>20.9 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from sh
Collecting slicer==0.0.7
  Downloading slicer-0.0.7-py3-none-any.whl (14 kB)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from sh
Requirement already satisfied: tqdm>4.25.0 in /usr/local/lib/python3.10/dist-packages (f
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (from s
Requirement already satisfied: cloudpickle in /usr/local/lib/python3.10/dist-packages (f
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (fr
Requirement already satisfied: llvmlite<0.40,>=0.39.0dev0 in /usr/local/lib/python3.10/d
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-pa
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from
Installing collected packages: slicer, shap
Successfully installed shap-0.41.0 slicer-0.0.7
```

```
import eli5
from eli5.sklearn import PermutationImportance


features = ['Months_on_book','Months_Inactive_12_mon','Contacts_Count_12_mon','Credit_Limit',
 'Total_Ct_Chng_Q4_Q1','0-DC','1-DC','2-DC','3-DC','4-DC','5-DC','College','Doctorate','Gradu
features
```

```
['Months_on_book',
 'Months_Inactive_12_mon',
 'Contacts_Count_12_mon',
 'Credit_Limit',
 'Total_Revolving_Bal',
 'Avg_Open_To_Buy',
 'Total_Amt_Chng_Q4_Q1',
 'Total_Trans_Amt',
 'Total_Trans_Ct',
 'Total_Ct_Chng_Q4_Q1',
 '0-DC',
 '1-DC',
 '2-DC',
 '3-DC',
 '4-DC',
 '5-DC',
 'College',
 'Doctorate',
 'Graduate',
 'High School',
 'Post-Graduate',
 'Uneducated',
 'Unknown',
 'Divorced',
 'Married',
 'Single',
 'Unknown',
 '$120K +',
 '$40K - $60K',
 '$60K - $80K',
 '$80K - $120K',
 'Less than $40K',
 'Unknown',
```

```
      'Blue',
      'Gold',
      'Platinum',
      'Silver',
      '1',
      '2',
      '3',
      '4',
      '5',
      '6']
```

```
perm = PermutationImportance(clf, random_state=1).fit(x_train.values, y_train)
```

```
eli5.show_weights(perm, feature_names = features)
```

Weight	Feature
0.2651 ± 0.0124	Total_Trans_Ct
0.1347 ± 0.0061	Total_Trans_Amt
0.0476 ± 0.0025	Total_Revolving_Bal
0.0343 ± 0.0022	Total_Ct_Chng_Q4_Q1
0.0325 ± 0.0032	6
0.0268 ± 0.0030	5
0.0258 ± 0.0028	4
0.0249 ± 0.0018	Months_Inactive_12_mon
0.0209 ± 0.0018	Credit_Limit
0.0199 ± 0.0024	Total_Amt_Chng_Q4_Q1
0.0197 ± 0.0029	3
0.0191 ± 0.0025	Months_on_book
0.0188 ± 0.0040	Avg_Open_To_Buy
0.0178 ± 0.0028	Less than $40K
0.0160 ± 0.0023	Contacts_Count_12_mon
0.0128 ± 0.0021	$40K - $60K
0.0119 ± 0.0010	$60K - $80K
0.0095 ± 0.0014	$80K - $120K
0.0090 ± 0.0011	Graduate
0.0088 ± 0.0022	Single

… 23 more …

KNN Model

```
import sklearn
from sklearn import neighbors

kncf = sklearn.neighbors.KNeighborsClassifier(n_neighbors=5, weights='uniform', p=2)

kn2 = kncf.fit(x_train.values, y_train)
```

Model Importance

```
kn3 = kncf.fit(x_train_2.values, y_train_2)
y_pred = kn3.predict(x_valid.values)

print (accuracy_score(y_valid,y_pred))

knn_acc = accuracy_score(y_valid,y_pred)
```

```
0.9086983343615053
```

Feature Importance

```
perm_kncf = PermutationImportance(kncf, random_state=1).fit(x_train.values, y_train)
```

```
eli5.show_weights(perm_kncf, feature_names = features)
```

Weight	Feature
0.0500 ± 0.0038	Total_Trans_Ct
0.0334 ± 0.0032	Total_Revolving_Bal
0.0206 ± 0.0034	Total_Ct_Chng_Q4_Q1
0.0130 ± 0.0033	Total_Trans_Amt
0.0119 ± 0.0023	Months_Inactive_12_mon
0.0086 ± 0.0014	Months_on_book
0.0083 ± 0.0020	Total_Amt_Chng_Q4_Q1
0.0060 ± 0.0027	Contacts_Count_12_mon
0.0046 ± 0.0016	Credit_Limit
0.0035 ± 0.0010	Avg_Open_To_Buy
0.0025 ± 0.0010	2
0.0016 ± 0.0009	1
0.0015 ± 0.0015	4
0.0009 ± 0.0006	Unknown
0.0005 ± 0.0008	Unknown
0.0005 ± 0.0013	Single
0.0003 ± 0.0013	2-DC
0.0002 ± 0.0009	$60K - $80K
0.0002 ± 0.0006	4-DC
0.0001 ± 0.0014	6
… 23 more …	

Linear Model

```
import sklearn
from sklearn import linear_model

lin = sklearn.linear_model.SGDClassifier(penalty='l2',alpha=0.0001)
lin2 = lin.fit(x_train.values,y_train)
```

Model Importance

```
lin3 = lin.fit(x_train_2.values, y_train_2)
y_pred = lin3.predict(x_valid.values)

print (accuracy_score(y_valid,y_pred))

lin_acc = accuracy_score(y_valid,y_pred)
```

```
0.905613818630475
```

Feature Importance

```
perm_lin = PermutationImportance(lin, random_state=1).fit(x_train.values, y_train)
```

```
eli5.show_weights(perm_lin, feature_names = features)
```

Weight	Feature
0.1722 ± 0.0061	Total_Trans_Ct
0.0511 ± 0.0030	Total_Trans_Amt
0.0295 ± 0.0041	Total_Revolving_Bal
0.0210 ± 0.0027	2
0.0187 ± 0.0022	Total_Ct_Chng_Q4_Q1
0.0158 ± 0.0025	Contacts_Count_12_mon
0.0153 ± 0.0035	Less than $40K
0.0147 ± 0.0021	1
0.0136 ± 0.0042	Single
0.0092 ± 0.0032	Months_Inactive_12_mon
0.0070 ± 0.0023	3-DC
0.0062 ± 0.0031	4-DC
0.0051 ± 0.0025	Married
0.0046 ± 0.0004	Blue
0.0045 ± 0.0008	2-DC
0.0043 ± 0.0016	Unknown
0.0035 ± 0.0012	Unknown
0.0023 ± 0.0016	Divorced
0.0023 ± 0.0009	Graduate
0.0022 ± 0.0009	3
… 23 more …	

Random Forest

```
!pip install deslib
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/pub
Collecting deslib
  Downloading DESlib-0.3.5-py3-none-any.whl (158 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 158.9/158.9 kB 9.3 MB/s eta 0:00:00
Requirement already satisfied: scikit-learn>=0.21.0 in /usr/local/lib/python3.10/dist-pa
Requirement already satisfied: numpy>=1.17.0 in /usr/local/lib/python3.10/dist-packages
```

```
Requirement already satisfied: scipy>=1.4.0 in /usr/local/lib/python3.10/dist-packages (
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-pa
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages
Installing collected packages: deslib
Successfully installed deslib-0.3.5
```

```
from sklearn.ensemble import RandomForestClassifier
from deslib.des.knora_e import KNORAE
pool_classifiers = RandomForestClassifier(n_estimators=10)
```

Model Importance

```
pool_classifiers.fit(x_train_2.values, y_train_2)
knorae = KNORAE(pool_classifiers)
knorae.fit(x_train_2.values, y_train_2)
y_pred_kn=knorae.predict(x_valid.values)
print (accuracy_score(y_valid,y_pred_kn))

ran_for_acc = accuracy_score(y_valid,y_pred_kn)
```

```
0.9426280074028378
```

Feature Importance

```
perm_kon = PermutationImportance(knorae, random_state=1).fit(x_train.values, y_train)


eli5.show_weights(perm_kon, feature_names = features)
```

Weight	Feature
0.1164 ± 0.0058	Total_Trans_Ct
0.0723 ± 0.0023	Total_Trans_Amt

Logistic Model

0.0096 ± 0.0013 2

```
from sklearn.linear_model import LogisticRegression
from sklearn import linear_model

log = LogisticRegression()
log.fit(x_train.values, y_train)
```

LogisticRegression

LogisticRegression()

0.0004 ± 0.0008 5

Model Importance

… 23 more …

```
log2 = log.fit(x_train_2.values, y_train_2)
y_pred = log2.predict(x_valid.values)

print (accuracy_score(y_valid,y_pred))

log_acc = accuracy_score(y_valid,y_pred)
```

```
0.9080814312152992
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: Converger
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Feature Importance

```
perm_log = PermutationImportance(log, random_state=1).fit(x_train.values, y_train)


eli5.show_weights(perm_log, feature_names = features)
```

Weight	Feature
0.1757 ± 0.0078	Total_Trans_Ct
0.0565 ± 0.0055	Total_Trans_Amt
0.0249 ± 0.0042	Total_Revolving_Bal
0.0165 ± 0.0037	Total_Ct_Chng_Q4_Q1
0.0107 ± 0.0012	Contacts_Count_12_mon
0.0093 ± 0.0029	2
0.0081 ± 0.0014	Months_Inactive_12_mon
0.0062 ± 0.0012	1
0.0034 ± 0.0019	5
0.0032 ± 0.0022	4
0.0032 ± 0.0016	6
0.0014 ± 0.0013	Married
0.0013 ± 0.0020	Less than $40K
0.0010 ± 0.0006	0-DC
0.0007 ± 0.0011	4-DC
0.0006 ± 0.0011	$60K - $80K
0.0004 ± 0.0005	Unknown

ADA Boosted Model

… 23 more …

```
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn import datasets

from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.model_selection import train_test_split, cross_val_score
```

Model and Model Importance

```
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier


ada_clf = AdaBoostClassifier(
    DecisionTreeClassifier(max_depth=1), n_estimators=3000,
    algorithm="SAMME.R", learning_rate=0.5)
ada_clf.fit(x_train.values, y_train)

y_pred=ada_clf.predict(x_valid.values)

print (accuracy_score(y_valid,y_pred))

ada_acc = accuracy_score(y_valid,y_pred)
```

```
0.9796421961752005
```

Feature Imporatnce

Do not run this again

```
#perm_ada = PermutationImportance(ada_clf, random_state=1).fit(x_train.values, y_train)
```

```
#eli5.show_weights(perm_ada, feature_names = features)
```

Gradient Boosting

```
from sklearn.ensemble import GradientBoostingClassifier
```

```
gb_clf=GradientBoostingClassifier(learning_rate=0.1, n_estimators=100, subsample=1, max_depth
```

Model Importance

```
gb_clf.fit(x_train.values, y_train)

y_pred_gb = gb_clf.predict(x_valid.values)
print (accuracy_score(y_valid,y_pred))

grad_acc = accuracy_score(y_valid,y_pred)
```

```
0.9796421961752005
```

Feature Importance

```
perm_grad = PermutationImportance(gb_clf, random_state=1).fit(x_train.values, y_train)
```

```
eli5.show_weights(perm_grad, feature_names = features)
```

Weight	Feature
0.0769 ± 0.0080	Total_Trans_Ct
0.0469 ± 0.0027	Total_Revolving_Bal
0.0178 ± 0.0024	Total_Trans_Amt
0.0153 ± 0.0019	Total_Ct_Chng_Q4_Q1
0.0053 ± 0.0019	Months_Inactive_12_mon
0.0023 ± 0.0010	Contacts_Count_12_mon
0.0022 ± 0.0006	Total_Amt_Chng_Q4_Q1
0.0010 ± 0.0008	2
0.0002 ± 0.0003	1
0 ± 0.0000	3-DC

Model Ranking

```
models = pd.DataFrame({
    'Model': ['clf','KNN','Linear','Random Forest','Logistic','Ada Boost','Gradient Boost'],
    'Score': [neur_acc, knn_acc, lin_acc, ran_for_acc,log_acc,ada_acc,grad_acc]})
models.sort_values(by='Score', ascending=False)
```

	Model	Score
5	Ada Boost	0.979642
6	Gradient Boost	0.979642
3	Random Forest	0.942628
0	clf	0.924738
1	KNN	0.908698
4	Logistic	0.908081
2	Linear	0.905614

Demographic Only Model

One-Hot Encode Dependent Count, Education Level, and Marital Status

```
encode_DC=OneHotEncoder(sparse=False)
mat_DC=encode_DC.fit_transform(train_df["Dependent_count"].to_numpy().reshape(-1,1))
df_DC0=pd.DataFrame(mat_DC,columns=encode_DC.categories_)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWa
  warnings.warn(
```

```
encode_EL=OneHotEncoder(sparse=False)
mat_EL=encode_EL.fit_transform(train_df["Education_Level"].to_numpy().reshape(-1,1))
df_EL0=pd.DataFrame(mat_EL,columns=encode_EL.categories_)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWa
  warnings.warn(
```

```
encode_MarS=OneHotEncoder(sparse=False)
mat_MarS=encode_MarS.fit_transform(train_df["Marital_Status"].to_numpy().reshape(-1,1))
df_MarS0=pd.DataFrame(mat_MarS,columns=encode_MarS.categories_)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWa
  warnings.warn(
```

```
credit_train_final_1 = pd.concat([df_DC0,df_EL0,df_MarS0], axis=1)
```

```
encode_DC=OneHotEncoder(sparse=False)
mat_DC=encode_DC.fit_transform(test_df["Dependent_count"].to_numpy().reshape(-1,1))
df_DC3=pd.DataFrame(mat_DC,columns=encode_DC.categories_)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWa
  warnings.warn(
```

```
encode_EL=OneHotEncoder(sparse=False)
mat_EL=encode_EL.fit_transform(test_df["Education_Level"].to_numpy().reshape(-1,1))
df_EL3=pd.DataFrame(mat_EL,columns=encode_EL.categories_)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWa
  warnings.warn(
```

```
encode_MarS=OneHotEncoder(sparse=False)
mat_MarS=encode_MarS.fit_transform(test_df["Marital_Status"].to_numpy().reshape(-1,1))
df_MarS3=pd.DataFrame(mat_MarS,columns=encode_MarS.categories_)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWa
  warnings.warn(
```

```
credit_test_final_1 = pd.concat([df_DC3,df_EL3,df_MarS3], axis=1)
```

```
y_train = train_df['Attrition_Flag']
```

```
x_train = credit_train_final_1
```

```
y_test = credit_test_final_1
```

Classifier

```
x_train_DO, x_valid_DO, y_train_DO, y_valid_DO = train_test_split(x_train, y_train, test_size
```

```
from sklearn.neural_network import MLPClassifier
clf = MLPClassifier(solver='adam', alpha=1e-5, random_state=1, max_iter=500,hidden_layer_size
```

```
clf.fit(x_train.values, y_train)
```

```
Iteration 1, loss = 0.80027409
Iteration 2, loss = 0.61880584
Iteration 3, loss = 0.50801749
Iteration 4, loss = 0.45585841
Iteration 5, loss = 0.44542038
Iteration 6, loss = 0.44198143
Iteration 7, loss = 0.43980868
Iteration 8, loss = 0.43811269
Iteration 9, loss = 0.43687784
Iteration 10, loss = 0.43611253
```

```
clf1 = clf.fit(x_train_DO.values, y_train_DO)
y_pred = clf1.predict(x_valid_DO.values)

print (accuracy_score(y_valid_DO,y_pred))

neur_acc = accuracy_score(y_valid_DO,y_pred)
```

```
Iteration 79, loss = 0.42875134
Iteration 80, loss = 0.42868566
Iteration 81, loss = 0.42859108
Iteration 82, loss = 0.42866602
Iteration 83, loss = 0.42845777
Iteration 84, loss = 0.42847185
Iteration 85, loss = 0.42838011
Iteration 86, loss = 0.42836739
Iteration 87, loss = 0.42828016
Iteration 88, loss = 0.42830251
Iteration 89, loss = 0.42822911
Iteration 90, loss = 0.42820549
Iteration 91, loss = 0.42810846
Iteration 92, loss = 0.42810432
Iteration 93, loss = 0.42806504
Iteration 94, loss = 0.42807216
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopp
```

Permutation Importance

```
features = ['0-DC','1-DC','2-DC','3-DC','4-DC','5-DC','College','Doctorate','Graduate','High
features
```

```
['0-DC',
 '1-DC',
 '2-DC',
 '3-DC',
 '4-DC',
 '5-DC',
 'College',
 'Doctorate',
 'Graduate',
 'High School',
 'Post-Graduate',
 'Uneducated',
 'Unknown',
 'Divorced',
 'Married',
 'Single',
 'Unknown']
```

```
perm = PermutationImportance(clf, random_state=1).fit(x_train_DO.values, y_train_DO)
```

```
eli5.show_weights(perm, feature_names = features)
```

```
/usr/local/lib/python3.10/dist-packages/eli5/formatters/html.py:233: RuntimeWarning: inv
  rel_weight = (abs(weight) / weight_range) ** 0.7
```

Weight	Feature
0 ± 0.0000	Unknown
0 ± 0.0000	Doctorate
0 ± 0.0000	1-DC
0 ± 0.0000	2-DC
0 ± 0.0000	3-DC
0 ± 0.0000	4-DC
0 ± 0.0000	5-DC
0 ± 0.0000	College
0 ± 0.0000	Graduate
0 ± 0.0000	Single
0 ± 0.0000	High School
0 ± 0.0000	Post-Graduate

Year 3 Classifier

```
MOB_36 = pd.DataFrame(train_df.where(train_df['Months_on_book'] == 36)).dropna()
MOB_36.head()
MOB_36.shape
```

```
(1968, 20)
```

```
encode_DC=OneHotEncoder(sparse=False)
mat_DC=encode_DC.fit_transform(MOB_36["Dependent_count"].to_numpy().reshape(-1,1))
df_DC_36=pd.DataFrame(mat_DC,columns=encode_DC.categories_)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWa
  warnings.warn(
```

```
encode_EL=OneHotEncoder(sparse=False)
mat_EL=encode_EL.fit_transform(MOB_36["Education_Level"].to_numpy().reshape(-1,1))
df_EL_36=pd.DataFrame(mat_EL,columns=encode_EL.categories_)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWa
  warnings.warn(
```

```
encode_MarS=OneHotEncoder(sparse=False)
mat_MarS=encode_MarS.fit_transform(MOB_36["Marital_Status"].to_numpy().reshape(-1,1))
df_MarS_36=pd.DataFrame(mat_MarS,columns=encode_MarS.categories_)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWa
  warnings.warn(
```

```
encode_IC=OneHotEncoder(sparse=False)
mat_IC=encode_IC.fit_transform(MOB_36["Income_Category"].to_numpy().reshape(-1,1))
df_IC_36=pd.DataFrame(mat_IC,columns=encode_IC.categories_)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWa
  warnings.warn(
```

```
encode_CC=OneHotEncoder(sparse=False)
mat_CC=encode_CC.fit_transform(MOB_36["Card_Category"].to_numpy().reshape(-1,1))
df_CC_36=pd.DataFrame(mat_CC,columns=encode_CC.categories_)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWa
  warnings.warn(
```

```
encode_TRC=OneHotEncoder(sparse=False)
mat_TRC=encode_TRC.fit_transform(MOB_36["Total_Relationship_Count"].to_numpy().reshape(-1,1))
df_TRC_36=pd.DataFrame(mat_TRC,columns=encode_TRC.categories_)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWa
  warnings.warn(
```

```
df_cats_36=pd.concat([df_DC_36,df_EL_36,df_MarS_36,df_IC_36,df_CC_36, df_TRC_36], axis=1)
df_cats_36
```

	0.0	1.0	2.0	3.0	4.0	5.0	College	Doctorate	Graduate	High School	...	Blue	Gold

```
num_pipeline = Pipeline([
        ('imputer', SimpleImputer(strategy="median")),
        ('std_scaler', StandardScaler()),
    ])
```

```
credit_test_continuous_36= num_pipeline.fit_transform(MOB_36[['Months_on_book','Months_Inacti
credit_test_continuous_36_1=pd.DataFrame(credit_test_continuous_36,columns=['Months_on_book',
```

```
credit_test_final_36=pd.concat([credit_test_continuous_36_1,df_cats_36],axis=1)
credit_test_final_36.shape
```

```
(1968, 43)
```

1965	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0

```
y_train_36 = MOB_36['Attrition_Flag']
```

```
x_train_36 = credit_test_final_36
```

1968 rows × 43 columns

```
x_train_36, x_valid_36, y_train_36, y_valid_36 = train_test_split(x_train_36, y_train_36, tes
```

```
from sklearn.neural_network import MLPClassifier
clf_36 = MLPClassifier(solver='adam', alpha=1e-5, random_state=1, max_iter=500,hidden_layer_s

clf_36.fit(x_train_36.values, y_train_36)
```

```
Iteration 450, loss = 0.01968604
Iteration 451, loss = 0.01967066
Iteration 452, loss = 0.01954010
Iteration 453, loss = 0.01945490
Iteration 454, loss = 0.01923714
Iteration 455, loss = 0.01910759
Iteration 456, loss = 0.01904511
Iteration 457, loss = 0.01892416
Iteration 458, loss = 0.01888810
Iteration 459, loss = 0.01870968
Iteration 460, loss = 0.01861669
Iteration 461, loss = 0.01871480
Iteration 462, loss = 0.01825258
Iteration 463, loss = 0.01825399
Iteration 464, loss = 0.01802318
Iteration 465, loss = 0.01793963
Iteration 466, loss = 0.01777683
Iteration 467, loss = 0.01762796
Iteration 468, loss = 0.01747180
Iteration 469, loss = 0.01737857
Iteration 470, loss = 0.01723018
Iteration 471, loss = 0.01713329
Iteration 472, loss = 0.01714166
Iteration 473, loss = 0.01688599
Iteration 474, loss = 0.01704957
Iteration 475, loss = 0.01678567
Iteration 476, loss = 0.01653030
Iteration 477, loss = 0.01646102
Iteration 478, loss = 0.01636600
Iteration 479, loss = 0.01627132
Iteration 480, loss = 0.01604380
Iteration 481, loss = 0.01602434
Iteration 482, loss = 0.01588516
Iteration 483, loss = 0.01569634
Iteration 484, loss = 0.01562575
Iteration 485, loss = 0.01557805
Iteration 486, loss = 0.01543866
Iteration 487, loss = 0.01531417
Iteration 488, loss = 0.01522132
Iteration 489, loss = 0.01508332
Iteration 490, loss = 0.01501114
Iteration 491, loss = 0.01490207
Iteration 492, loss = 0.01475689
Iteration 493, loss = 0.01468635
Iteration 494, loss = 0.01454602
Iteration 495, loss = 0.01450902
Iteration 496, loss = 0.01440549
Iteration 497, loss = 0.01428153
Iteration 498, loss = 0.01416403
Iteration 499, loss = 0.01403508
Iteration 500, loss = 0.01395054
/usr/local/lib/python3.10/dist-packages/sklearn/neural_network/_multilayer_perceptron
  warnings.warn(
```

▾ MLPClassifier

```
MLPClassifier(alpha=1e-05, hidden_layer_sizes=(13, 6), max_iter=500,
              random_state=1, verbose=True)
```

```
clf1 = clf_36.fit(x_train_36.values, y_train_36)
y_pred = clf1.predict(x_valid_36.values)

print (accuracy_score(y_valid_36,y_pred))

neur_acc = accuracy_score(y_valid_36,y_pred)
```

```
Iteration 495, loss = 0.01450902
Iteration 496, loss = 0.01440549
Iteration 497, loss = 0.01428153
Iteration 498, loss = 0.01416403
Iteration 499, loss = 0.01403508
Iteration 500, loss = 0.01395054
0.8807106598984772
/usr/local/lib/python3.10/dist-packages/sklearn/neural_network/_multilayer_perceptron
```

```
import eli5
from eli5.sklearn import PermutationImportance
```

```
features = ['Months_on_book','Months_Inactive_12_mon','Contacts_Count_12_mon','Credit_Limit',
 'Total_Ct_Chng_Q4_Q1','0-DC','1-DC','2-DC','3-DC','4-DC','5-DC','College','Doctorate','Gradu
features
```

```
['Months_on_book',
 'Months_Inactive_12_mon',
 'Contacts_Count_12_mon',
 'Credit_Limit',
 'Total_Revolving_Bal',
 'Avg_Open_To_Buy',
 'Total_Amt_Chng_Q4_Q1',
 'Total_Trans_Amt',
 'Total_Trans_Ct',
 'Total_Ct_Chng_Q4_Q1',
 '0-DC',
 '1-DC',
 '2-DC',
 '3-DC',
 '4-DC',
 '5-DC',
 'College',
 'Doctorate',
 'Graduate',
 'High School',
 'Post-Graduate',
 'Uneducated',
 'Unknown',
 'Divorced',
 'Married',
 'Single',
 'Unknown',
 '$120K +',
 '$40K - $60K',
 '$60K - $80K',
 '$80K - $120K',
 'Less than $40K',
 'Unknown',
 'Blue',
 'Gold',
 'Platinum',
 'Silver',
```

```
'1',
'2',
'3',
'4',
'5',
'6']
```

```
perm = PermutationImportance(clf_36, random_state=1).fit(x_train_36.values, y_train_36)
```

```
eli5.show_weights(perm, feature_names = features)
```

Weight	Feature
0.2759 ± 0.0158	Total_Trans_Ct
0.1488 ± 0.0038	Total_Trans_Amt
0.0748 ± 0.0085	Total_Revolving_Bal
0.0653 ± 0.0060	Total_Ct_Chng_Q4_Q1
0.0537 ± 0.0047	Months_Inactive_12_mon
0.0332 ± 0.0041	3
0.0285 ± 0.0060	4
0.0278 ± 0.0054	6
0.0274 ± 0.0044	Contacts_Count_12_mon
0.0260 ± 0.0032	Total_Amt_Chng_Q4_Q1
0.0253 ± 0.0049	5
0.0249 ± 0.0058	Credit_Limit
0.0216 ± 0.0029	$60K - $80K
0.0213 ± 0.0072	Avg_Open_To_Buy
0.0179 ± 0.0048	Less than $40K
0.0166 ± 0.0041	Married
0.0161 ± 0.0076	Graduate
0.0156 ± 0.0025	2-DC
0.0155 ± 0.0026	$80K - $120K
0.0149 ± 0.0019	0-DC

… 23 more …

Correlation Matrix

```
encode_AF=OneHotEncoder(sparse=False)
mat_AF=encode_AF.fit_transform(MOB_36["Attrition_Flag"].to_numpy().reshape(-1,1))
df_AF_36=pd.DataFrame(mat_AF,columns=encode_AF.categories_)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWa
  warnings.warn(
```

```
df_cats_CM_36=pd.concat([df_DC_36,df_EL_36,df_MarS_36,df_IC_36,df_CC_36,df_TRC_36,df_AF_36],
```

```
num_pipeline = Pipeline([
        ('imputer', SimpleImputer(strategy="median")),
```

```
        ('std_scaler', StandardScaler()),
    ])


credit_train_continuous_36= num_pipeline.fit_transform(train_df[['Months_on_book','Months_Ina
credit_train_continuous_36= pd.DataFrame(credit_train_continuous,columns=['Months_on_book','M


correlation_train_final_36=pd.concat([credit_train_continuous_36,df_cats_CM_36],axis=1)


corr_mat_36 = correlation_train_final_36.corr()


corr_mat_36
```

	Months_on_book	Months_Inactive_12_mon	Contacts_Count_12_mo
Months_on_book	1.000000	0.070232	-0.00686
Months_Inactive_12_mon	0.070232	1.000000	0.02400
Contacts_Count_12_mon	-0.006860	0.024000	1.00000
Credit_Limit	0.000296	-0.021909	0.02620
Total_Revolving_Bal	0.009110	-0.039588	-0.05727
Avg_Open_To_Buy	-0.000526	-0.018327	0.03135
Total_Amt_Chng_Q4_Q1	-0.054831	-0.022899	-0.02111
Total_Trans_Amt	-0.034397	-0.027393	-0.10806
Total_Trans_Ct	-0.046780	-0.030351	-0.14998
Total_Ct_Chng_Q4_Q1	-0.012581	-0.032113	-0.09090
(0.0,)	-0.040499	-0.008249	-0.01283
(1.0,)	-0.004899	0.031242	-0.01446
(2.0,)	-0.031415	-0.021972	-0.03858
(3.0,)	0.039683	-0.024907	0.01303
(4.0,)	-0.011290	0.017938	0.03614
(5.0,)	0.065749	0.021352	0.03134
(College,)	0.026433	0.008939	-0.00457
(Doctorate,)	-0.001303	0.018004	0.02145
(Graduate,)	-0.001557	-0.011031	-0.01458
(High School,)	0.028688	0.014072	0.02536
(Post-Graduate,)	-0.016462	-0.002854	-0.00586
(Uneducated,)	-0.035896	-0.012628	-0.03199
(Unknown,)	-0.004730	-0.004939	0.01715
(Divorced,)	0.021319	-0.000153	-0.01554
(Married,)	-0.039774	-0.004081	-0.01244
(Single,)	0.034722	0.004635	0.00972

Best Model

Check for overfitting and underfitting

Run a grid search to refine model

Early Stopping-

Boosted models are prone to overfitting

```
from sklearn.metrics import log_loss

ada_clf = AdaBoostClassifier(
    DecisionTreeClassifier(max_depth=1), n_estimators=3000,
    algorithm="SAMME.R", learning_rate=0.5)
min_val_error = 0
error_going_up = 0

for n_estimators in range(1, 120):
    ada_clf.n_estimators = n_estimators
    ada_clf.fit(x_train_2.values, y_train_2)
    y_pred = ada_clf.predict(x_valid.values)
    pred = ada_clf.predict_proba(x_valid.values)
    val_error = log_loss(y_valid, pred)
    print(val_error)
    if val_error > min_val_error:
        min_val_error = val_error
        error_going_up = 0
    else:
        error_going_up += 1
        if error_going_up == 5:
            break
```

```
0.3355088276815782
0.3243994326106479
0.3483982469976743
0.36981987249843634
0.4047527671130201
0.4193386047904828
0.4368093006158291
0.4550613522700805
0.47225022759025337
0.4876031992914211
0.49201809426447957
0.5061593959159203
0.5071291780264979
0.5199566420471958
0.5272561498300325
0.5287963928004983
0.5375569284208208
0.5424150677398495
0.5457957312628073
0.5533181942164842
0.5568464849569139
0.5572882871639228
```

```
0.5627449426614926
0.5624005957384972
0.5645788488129984
0.5687284294599608
0.57200417041992
0.5763439332433833
0.5752689202096072
0.5800224596345224
0.582353896494676
0.5536264458129334
0.5575729918592979
0.5604478297942462
0.5604849186447206
0.5644433400596117
```

Grid Search

```
from sklearn.model_selection import GridSearchCV

param_grid = {'max_features': ['auto', 'sqrt', 'log2'],'max_depth':[1,2,3]}

ada_clf = DecisionTreeClassifier()

grid_search=GridSearchCV(ada_clf,param_grid, cv=3,return_train_score=True, scoring='accuracy'

grid_search.fit(x_train_2.values,y_train_2)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning: `ma
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning: `ma
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning: `ma
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning: `ma
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning: `ma
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning: `ma
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning: `ma
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning: `ma
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning: `ma
  warnings.warn(
```

GridSearchCV

estimator: DecisionTreeClassifier

DecisionTreeClassifier

```
cvres = grid_search.cv_results_
for mean_score, params in zip(cvres["mean_test_score"], cvres["params"]):
  print((mean_score), params)
```

```
0.8452401865848572 {'max_depth': 1, 'max_features': 'auto'}
0.8435426557206597 {'max_depth': 1, 'max_features': 'sqrt'}
0.8435426557206597 {'max_depth': 1, 'max_features': 'log2'}
0.8489432475819951 {'max_depth': 2, 'max_features': 'auto'}
0.8438512976959683 {'max_depth': 2, 'max_features': 'sqrt'}
0.8578886518015779 {'max_depth': 2, 'max_features': 'log2'}
0.8591228626436093 {'max_depth': 3, 'max_features': 'auto'}
0.8637570626310408 {'max_depth': 3, 'max_features': 'sqrt'}
0.8566573688450134 {'max_depth': 3, 'max_features': 'log2'}
```

```
grid_search.best_params_
```

```
{'max_depth': 3, 'max_features': 'sqrt'}
```

```
grid_search.best_estimator_
```

```
DecisionTreeClassifier
DecisionTreeClassifier(max_depth=3, max_features='sqrt')
```

Grid Search 2- Did not like the above results

```
from sklearn.ensemble import RandomForestClassifier
from deslib.des.knora_e import KNORAE
pool_classifiers = RandomForestClassifier()
```

```
pool_classifiers.fit(x_train_2.values, y_train_2)
knorae = KNORAE(pool_classifiers)
knorae.fit(x_train_2.values, y_train_2)
y_pred_kn=knorae.predict(x_valid.values)
print (accuracy_score(y_valid,y_pred_kn))
```

```
0.9487970388648982
```

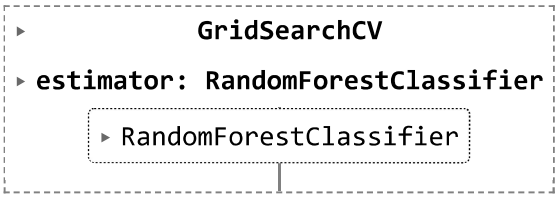
```
from sklearn.ensemble import RandomForestClassifier
from deslib.des.knora_e import KNORAE
pool_classifiers = RandomForestClassifier()
```

```
pool_classifiers.fit(x_train_2.values, y_train_2)
```

```
y_pred_kn=pool_classifiers.predict(x_valid.values)
print (accuracy_score(y_valid,y_pred_kn))
```

```
0.9487970388648982
```

```
from sklearn.model_selection import GridSearchCV

param_grid = {'n_estimators':[100,150],'n_jobs':[1,2,3],'criterion':['gini', 'entropy', 'log_

grid_search=GridSearchCV(pool_classifiers,param_grid, cv=3,return_train_score=True, scoring='

grid_search.fit(x_train_2.values,y_train_2)
```




```
cvres = grid_search.cv_results_
for mean_score, params in zip(cvres["mean_test_score"], cvres["params"]):
  print((mean_score), params)
```

```
0.9486190378254237 {'criterion': 'gini', 'n_estimators': 100, 'n_jobs': 1}
0.9483103244382743 {'criterion': 'gini', 'n_estimators': 100, 'n_jobs': 2}
0.9459966522129101 {'criterion': 'gini', 'n_estimators': 100, 'n_jobs': 3}
0.9469222210796326 {'criterion': 'gini', 'n_estimators': 150, 'n_jobs': 1}
0.9489281082717763 {'criterion': 'gini', 'n_estimators': 150, 'n_jobs': 2}
0.9473854696899583 {'criterion': 'gini', 'n_estimators': 150, 'n_jobs': 3}
0.9463052941882187 {'criterion': 'entropy', 'n_estimators': 100, 'n_jobs': 1}
0.9470761135962431 {'criterion': 'entropy', 'n_estimators': 100, 'n_jobs': 2}
0.9480026108168943 {'criterion': 'entropy', 'n_estimators': 100, 'n_jobs': 3}
0.9475392907947281 {'criterion': 'entropy', 'n_estimators': 150, 'n_jobs': 1}
0.9478474328871522 {'criterion': 'entropy', 'n_estimators': 150, 'n_jobs': 2}
0.9478476471226741 {'criterion': 'entropy', 'n_estimators': 150, 'n_jobs': 3}
0.9486192520609457 {'criterion': 'log_loss', 'n_estimators': 100, 'n_jobs': 1}
0.95062521066493 {'criterion': 'log_loss', 'n_estimators': 100, 'n_jobs': 2}
0.9469222924914735 {'criterion': 'log_loss', 'n_estimators': 100, 'n_jobs': 3}
0.9469220782559514 {'criterion': 'log_loss', 'n_estimators': 150, 'n_jobs': 1}
0.9447620128998349 {'criterion': 'log_loss', 'n_estimators': 150, 'n_jobs': 2}
0.9469221496677921 {'criterion': 'log_loss', 'n_estimators': 150, 'n_jobs': 3}
```

```
grid_search.best_params_
```

```
{'criterion': 'log_loss', 'n_estimators': 100, 'n_jobs': 2}
```

```
grid_search.best_estimator_
```

RandomForestClassifier

Shap Model

```
import shap
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPRegressor
from sklearn.pipeline import make_pipeline

from sklearn.neural_network import MLPClassifier
clf = MLPClassifier(solver='adam', alpha=1e-5, random_state=1, max_iter=500,hidden_layer_size

clf.fit(x_train.values, y_train)
```

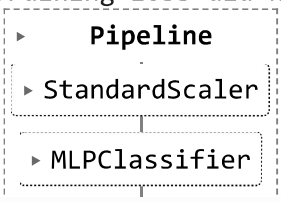
```
Iteration 1, loss = 0.80027409
Iteration 2, loss = 0.61880584
Iteration 3, loss = 0.50801749
Iteration 4, loss = 0.45585841
Iteration 5, loss = 0.44542038
Iteration 6, loss = 0.44198143
Iteration 7, loss = 0.43980868
Iteration 8, loss = 0.43811269
Iteration 9, loss = 0.43687784
```

```
model = make_pipeline(
    StandardScaler(),
    MLPClassifier(solver='adam', alpha=1e-5, random_state=1, max_iter=500,hidden_layer_sizes=
)
```

```
Iteration 16, loss = 0.43437607
```

```
model.fit(x_train_2.values,y_train_2)
```

```
Iteration 327, loss = 0.09122607
Iteration 328, loss = 0.09123128
Iteration 329, loss = 0.09093309
Iteration 330, loss = 0.09089071
Iteration 331, loss = 0.09075677
Iteration 332, loss = 0.09060392
Iteration 333, loss = 0.09122256
Iteration 334, loss = 0.09068519
Iteration 335, loss = 0.09036972
Iteration 336, loss = 0.09040828
Iteration 337, loss = 0.09007723
Iteration 338, loss = 0.09042930
Iteration 339, loss = 0.09016750
Iteration 340, loss = 0.09044008
Iteration 341, loss = 0.09027346
Iteration 342, loss = 0.08986571
Iteration 343, loss = 0.08966714
Iteration 344, loss = 0.08968989
Iteration 345, loss = 0.08960423
Iteration 346, loss = 0.08953677
Iteration 347, loss = 0.08935481
Iteration 348, loss = 0.08943784
Iteration 349, loss = 0.08945032
Iteration 350, loss = 0.08929335
Iteration 351, loss = 0.08917775
Iteration 352, loss = 0.08914351
Iteration 353, loss = 0.08906014
Iteration 354, loss = 0.08913903
Iteration 355, loss = 0.08889203
Iteration 356, loss = 0.08866948
Iteration 357, loss = 0.08841754
Iteration 358, loss = 0.08823337
Iteration 359, loss = 0.08869320
Iteration 360, loss = 0.08830495
Iteration 361, loss = 0.08838625
Iteration 362, loss = 0.08862849
Iteration 363, loss = 0.08867176
Iteration 364, loss = 0.08795586
Iteration 365, loss = 0.08758091
Iteration 366, loss = 0.08785102
Iteration 367, loss = 0.08770253
Iteration 368, loss = 0.08768283
Iteration 369, loss = 0.08784202
Iteration 370, loss = 0.08760805
Iteration 371, loss = 0.08800575
Iteration 372, loss = 0.08815272
Iteration 373, loss = 0.08801501
Iteration 374, loss = 0.08791364
Iteration 375, loss = 0.08759710
Iteration 376, loss = 0.08781436
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopp
```



```
y_pred=model.predict(x_valid.values)


model.score(x_valid.values,y_valid)

     0.9148673658235656


from sklearn.ensemble import RandomForestRegressor
from sklearn import linear_model

nuer = MLPClassifier(solver='adam', alpha=1e-5, random_state=1, max_iter=500,hidden_layer_siz
nuer.fit(x_train_2.values, y_train_2)
print("Score from random forest")
print(nuer.score(x_valid.values,y_valid))

# explain all the predictions in the test set
explainer = shap.Explainer(nuer.predict,x_train_2.values)
```

```
Iteration 397, loss = 0.07191773
Iteration 398, loss = 0.07302157
Iteration 399, loss = 0.07228734
Iteration 400, loss = 0.07178416
Iteration 401, loss = 0.07185597
Iteration 402, loss = 0.07191786
Iteration 403, loss = 0.07107665
Iteration 404, loss = 0.07138789
Iteration 405, loss = 0.07237895
Iteration 406, loss = 0.07252019
Iteration 407, loss = 0.07192169
Iteration 408, loss = 0.07164275
Iteration 409, loss = 0.07119523
Iteration 410, loss = 0.07161371
Iteration 411, loss = 0.07186499
Iteration 412, loss = 0.07111977
Iteration 413, loss = 0.07217954
Iteration 414, loss = 0.07100689
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopp
Score from random forest
```

Correlation Matrix

```
encode_AF=OneHotEncoder(sparse=False)
mat_AF=encode_AF.fit_transform(train_df["Attrition_Flag"].to_numpy().reshape(-1,1))
df_AF=pd.DataFrame(mat_AF,columns=encode_AF.categories_)
```

```
`sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `spar
```

```
df_cats_CM=pd.concat([df_DC,df_EL,df_MarS,df_IC,df_CC,df_TRC,df_AF], axis=1)
```

```
num_pipeline = Pipeline([
        ('imputer', SimpleImputer(strategy="median")),
        ('std_scaler', StandardScaler()),
    ])
```

```
credit_train_continuous_2= num_pipeline.fit_transform(train_df[['Months_on_book','Months_Inac
credit_train_continuous_2=pd.DataFrame(credit_train_continuous,columns=['Months_on_book','Mon
```

```
correlation_train_final=pd.concat([credit_train_continuous_2,df_cats_CM],axis=1)
```

```
corr_mat = correlation_train_final.corr()
```

```
corr_mat
```

	Months_on_book	Months_Inactive_12_mon	Contacts_Count_12_mo
Months_on_book	1.000000	0.070232	-0.00686
Months_Inactive_12_mon	0.070232	1.000000	0.02400
Contacts_Count_12_mon	-0.006860	0.024000	1.00000
Credit_Limit	0.000296	-0.021909	0.02620
Total_Revolving_Bal	0.009110	-0.039588	-0.05727
Avg_Open_To_Buy	-0.000526	-0.018327	0.03135
Total_Amt_Chng_Q4_Q1	-0.054831	-0.022899	-0.02111
Total_Trans_Amt	-0.034397	-0.027393	-0.10806
Total_Trans_Ct	-0.046780	-0.030351	-0.14998
Total_Ct_Chng_Q4_Q1	-0.012581	-0.032113	-0.09090
(0,)	0.063117	-0.009043	0.02493
(1,)	0.067727	0.018435	0.01753
(2,)	0.002454	-0.002761	-0.00455
(3,)	-0.050575	0.003457	-0.00250
(4,)	-0.051839	-0.003673	-0.02143
(5,)	-0.021179	-0.017446	-0.01519
(College,)	-0.016421	0.006214	-0.01068
(Doctorate,)	0.026179	-0.002625	0.00657

```
print(corr_mat.iloc[44])
```

```
Months_on_book           -0.009151
Months_Inactive_12_mon   -0.141771
Contacts_Count_12_mon    -0.206131
Credit_Limit              0.023687
Total_Revolving_Bal       0.262974
Avg_Open_To_Buy          -0.000042
Total_Amt_Chng_Q4_Q1      0.131198
Total_Trans_Amt           0.168031
Total_Trans_Ct            0.369266
Total_Ct_Chng_Q4_Q1       0.290921
(0,)                      0.013986
(1,)                      0.014172
(2,)                      0.012018
(3,)                     -0.028729
(4,)                     -0.009323
(5,)                      0.006363
(College,)                0.005003
(Doctorate,)             -0.027273
```

```
(Graduate,)               0.006538
(High School,)            0.011778
(Post-Graduate,)         -0.005456
(Uneducated,)             0.000645
(Unknown,)               -0.007332
(Divorced,)               0.002576
(Married,)                0.019258
(Single,)                -0.020389
(Unknown,)               -0.001333
($120K +,)               -0.009684
($40K - $60K,)            0.015460
($60K - $80K,)            0.028925
($80K - $120K,)          -0.003852
(Less than $40K,)        -0.023579
(Unknown,)               -0.002343
(Blue,)                  -0.001339
(Gold,)                  -0.002669
(Platinum,)              -0.020790
(Silver,)                 0.006652
(1,)                     -0.077906
(2,)                     -0.112156
(3,)                     -0.022161
(4,)                      0.058473
(5,)                      0.049809
(6,)                      0.068133
(Attrited Customer,)     -1.000000
(Existing Customer,)      1.000000
Name: (Existing Customer,), dtype: float64
```

```
# "2," and "1," are number of contacts
# "3," is the number of dependents
c1 = corr_mat.iloc[43]
c1[c1 < 1].transpose().sort_values(ascending=False).drop_duplicates()
```

```
Contacts_Count_12_mon     0.206131
Months_Inactive_12_mon    0.141771
(2,)                      0.112156
(1,)                      0.077906
(3,)                      0.028729
(Doctorate,)              0.027273
(Less than $40K,)         0.023579
(3,)                      0.022161
(Platinum,)               0.020790
(Single,)                 0.020389
($120K +,)                0.009684
(4,)                      0.009323
Months_on_book            0.009151
(Unknown,)                0.007332
(Post-Graduate,)          0.005456
($80K - $120K,)           0.003852
(Gold,)                   0.002669
(Unknown,)                0.002343
(Blue,)                   0.001339
(Unknown,)                0.001333
Avg_Open_To_Buy           0.000042
```

```
(Uneducated,)            -0.000645
(Divorced,)              -0.002576
(College,)               -0.005003
(5,)                     -0.006363
(Graduate,)              -0.006538
(Silver,)                -0.006652
(High School,)           -0.011778
(2,)                     -0.012018
(0,)                     -0.013986
(1,)                     -0.014172
($40K - $60K,)           -0.015460
(Married,)               -0.019258
Credit_Limit             -0.023687
($60K - $80K,)           -0.028925
(5,)                     -0.049809
(4,)                     -0.058473
(6,)                     -0.068133
Total_Amt_Chng_Q4_Q1     -0.131198
Total_Trans_Amt          -0.168031
Total_Revolving_Bal      -0.262974
Total_Ct_Chng_Q4_Q1      -0.290921
Total_Trans_Ct           -0.369266
(Existing Customer,)     -1.000000
Name: (Attrited Customer,), dtype: float64
```

```
full_df.head(1)
```

	CLIENTNUM	Attrition_Flag	Customer_Age	Gender	Dependent_count	Education_Level	Ma
0	768805383	Existing Customer	45	M	3	High School	

1 rows × 23 columns

```
full_df[['Total_Relationship_Count','Attrition_Flag']].groupby(['Attrition_Flag']).count()
```

Attrition_Flag	Total_Relationship_Count
Attrited Customer	1627
Existing Customer	8500

```
sns.heatmap(corr_mat, annot=True)
plt.show()
```