Credit Card Prediction

Analysing Credit Card Details of the user to check whether the account is Risky or not.

By - Diwakar Singh
Profile – Machine Learning Intern
Batch Name: MIP-ML-09

Problems Statement

The primary objective of this project is to predict the approval or rejection of credit card applications. The challenge lies in understanding the key factors influencing credit card approval decisions and building a predictive model to assist in the decision-making procsss.

Project Objectives

- Exploratory Data Analysis (EDA)
- Feature Engineering
- Data Preprocessing
- ML Model Development
- Model Evaluation
- Predicting Credit Card Approval
- Recommendations

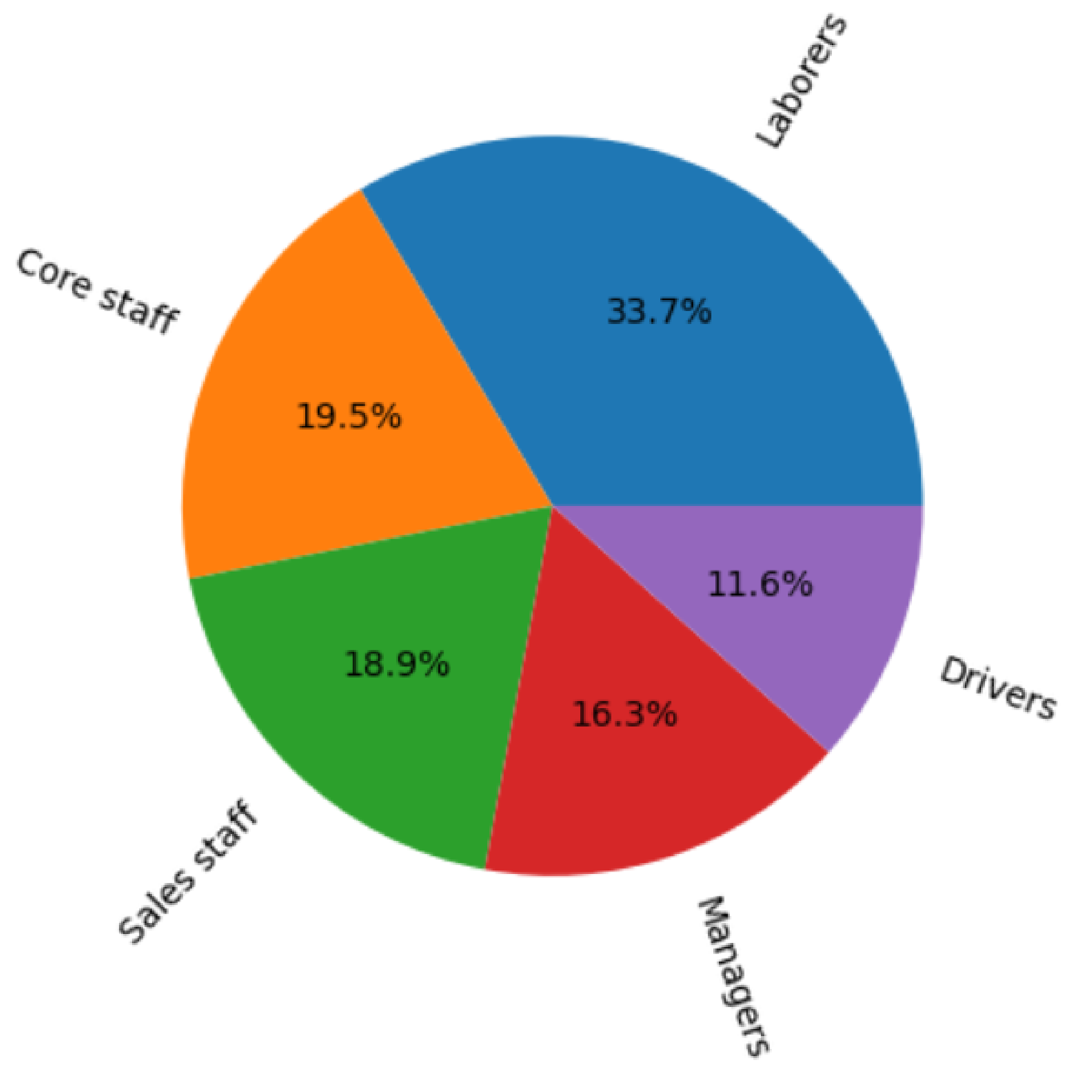
Data Exploration...

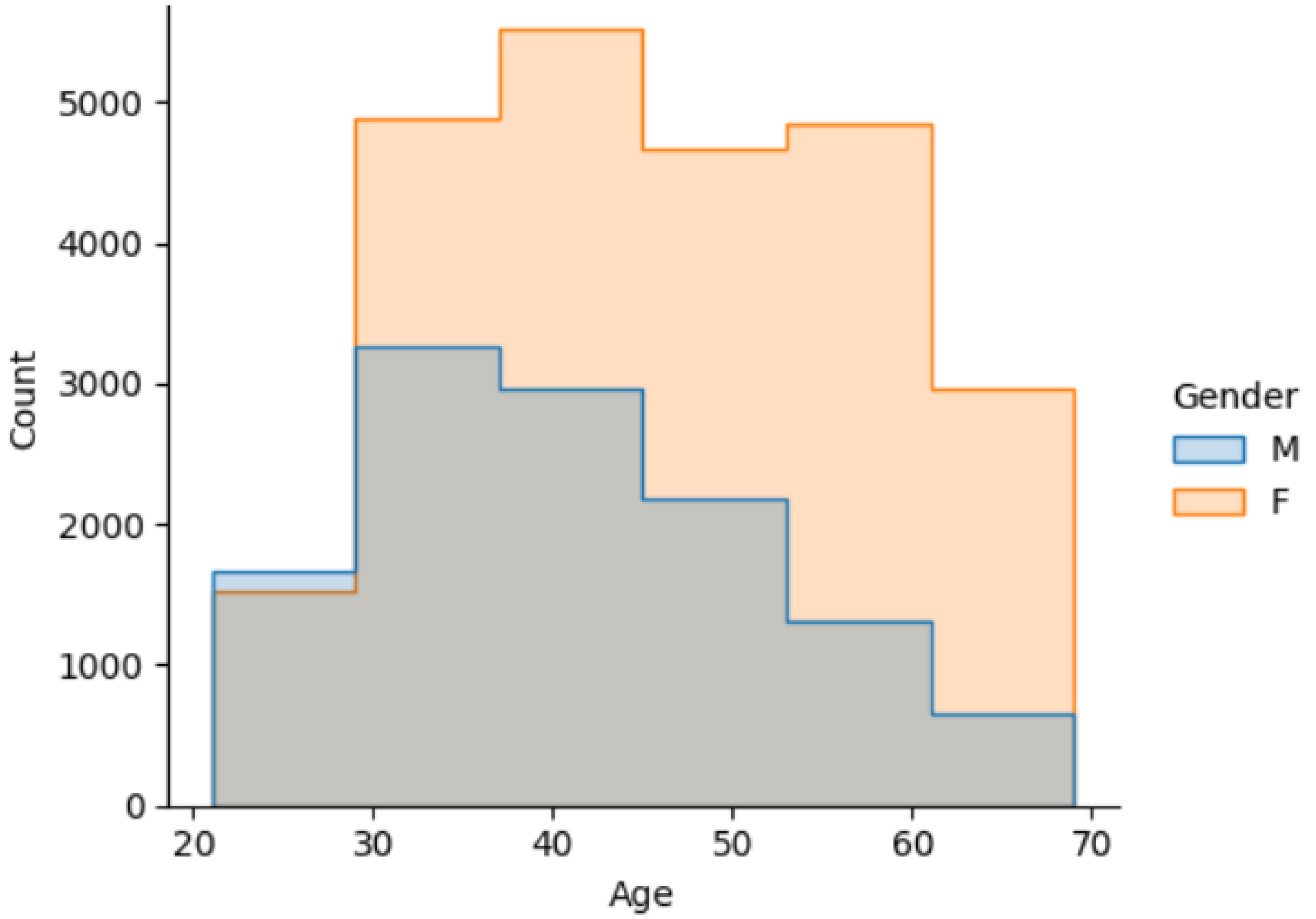
The very First step of Data Analysis: Viewing the data and finding the obvious patterns, and visualizing how out data actually looks like, how can use it to get good results.

	Gender	Has a car	Has a property	Children count	Income	Employment status	Education level	Marital status	Dwelling	Age	Employment length in years	Has a work phone	Has a phone	Has an email	Job title	Family member count	Account age
674	M	Y	Y	0	270000.0	Working	Higher education	Married	House / apartment	34	1	0	0	0	Other	2.0	31
440	M	Y	Y	0	135000.0	Working	Secondary / secondary special	Married	House / apartment	28	3	0	0	1	Sales staff	2.0	35
228	F	N	N	0	135000.0	Working	Higher education	Single / not married	House / apartment	38	1	1	0	0	Other	1.0	23
923	M	Y	N	1	99000.0	Pensioner	Secondary / secondary special	Married	House / apartment	51	-1	0	1	0	Unemployed	3.0	6
919	F	Y	Y	0	180000.0	Commercial associate	Secondary / secondary special	Married	House / apartment	37	7	0	1	0	Other	2.0	48

Exploratory Data Analysis (EDA)

Analysis of Data using Visual Representation of Graphs, finding trends and patterns in the data.





Feature Engineering

Feature engineering is shaping raw data into features that power machine learning models.

Feature Engineering

- Gender, Has a car, Has a property, Employment Status, Education level, Marital status, Dwelling, Job Title need to be OneHotEncoded.
- Income column needs to be normalized.
- We can remove Has a mobile phone column as it's all values are 1 which mainly means everyone has their phone, so there is not any need of this column.
- As there are very less values in some of the categories like in Dwelling Column there are around 5 categories but the data is not evenly distributed so just pick the top columns which has a greater values and then rename all the other categories as other. (Dwelling, Education level, Marital status, Job title).
- Remove Students Category from Employment Status column as it only has 7 enteries.
- Drop ID Column as it won't be affecting our result.

Has a mobile phone

[37]:
```python
# Removing Has a mobile phone column
trainDF.drop(columns= ["Has a mobile phone"], inplace= True)
```

Employment Status

[38]:
```python
# Removing Student from Employment Status column

trainDF["Employment status"].value_counts()
```

[38]:
```
Employment status
Working                 18819
Commercial associate     8490
Pensioner                6152
State servant            2985
Student                    11
Name: count, dtype: int64
```

[39]:
```python
trainDF = trainDF.loc[trainDF["Employment status"] != "Student"]
```

Education level

[40]:
```python
trainDF["Education level"].value_counts()
```

[40]:
```
Education level
Secondary / secondary special    24775
Higher education                  9855
Incomplete higher                 1410
```

[41]:
```python
# Merging Last 3 Categories as other

def renameEducationLevel(eduLevel):
    if eduLevel == "Incomplete higher":
        return "Other"
    elif eduLevel == "Lower secondary":
        return "Other"
    elif eduLevel == "Academic degree":
        return "Other"
    else:
        return eduLevel

trainDF["Education level"] = trainDF["Education level"].apply(renameEducationLevel)
```

[42]:
```python
trainDF["Education level"].value_counts()
```

[42]:
```
Education level
Secondary / secondary special    24775
Higher education                  9855
Other                             1816
Name: count, dtype: int64
```

Martial Status

[43]:
```python
trainDF["Marital status"].value_counts()
```

[43]:
```
Marital status
Married                 25040
Single / not married     4828
Civil marriage           2943
```

Data Pre-Processing

Data preprocessing is cleaning and preparing raw data for machine learning algorithms.

[11]:
```
# Statistical Insight of the dataset
trainDF.describe()
```

[11]:

	ID	Children count	Income	Age	Employment length	Has a mobile phone	Has a work phone	Has a phone	Has an email	Family member count	Account age
count	3.645700e+04	36457.000000	3.645700e+04	36457.000000	36457.000000	36457.0	36457.000000	36457.000000	36457.000000	36457.000000	36457.000000
mean	5.078227e+06	0.430315	1.866857e+05	-15975.173382	59262.935568	1.0	0.225526	0.294813	0.089722	2.198453	-26.164193
std	4.187524e+04	0.742367	1.017892e+05	4200.549944	137651.334859	0.0	0.417934	0.455965	0.285787	0.911686	16.501854
min	5.008804e+06	0.000000	2.700000e+04	-25152.000000	-15713.000000	1.0	0.000000	0.000000	0.000000	1.000000	-60.000000
25%	5.042028e+06	0.000000	1.215000e+05	-19438.000000	-3153.000000	1.0	0.000000	0.000000	0.000000	2.000000	-39.000000
50%	5.074614e+06	0.000000	1.575000e+05	-15563.000000	-1552.000000	1.0	0.000000	0.000000	0.000000	2.000000	-24.000000
75%	5.115396e+06	1.000000	2.250000e+05	-12462.000000	-408.000000	1.0	0.000000	1.000000	0.000000	3.000000	-12.000000
max	5.150487e+06	19.000000	1.575000e+06	-7489.000000	365243.000000	1.0	1.000000	1.000000	1.000000	20.000000	0.000000

[12]:
```
# Checking Correlation of Is high risk column with other columns
trainDF.drop(columns= ["Gender", "Has a car", "Has a property",
                       "Employment status", "Education level",
                       "Marital status", "Dwelling", "Job title"]).corr()[["Is high risk"]]
```

[12]:

	Is high risk
ID	0.015588
Children count	-0.000308
Income	-0.001057
Age	0.001478
Employment length	0.005664
Has a mobile phone	NaN
Has a work phone	0.005640
Has a phone	0.001585
Has an email	-0.002434
Family member count	-0.005660
Account age	-0.060215
Is high risk	1.000000

Preprocessing

```
* Job Title has 9027 missing values.
* The Columns which has thier values in categories like, Gender, Has a Car,
  Has a property, etc. should be convertod into binary format using
  OneHotEncoding for better results.
* Some Columns like, Age, Employment Length and Account Age has negative values.
* Income Values should be scaled.
* Check the number of categories of each column and then convert them
  into category datatype.
```

Age

The age column need to be preprocessed and it's values should be changed before doing any analysis on it.

[13]:
```
# Todays date
today = date.today()

# Creating an function to modify the Age Column values.
def Age(age):
    daysBack = int(age)
    dob = (today + timedelta(days= daysBack))
    return today.year - dob.year

# Modifying Age column values to how old they are in years.
trainDF["Age"] = trainDF["Age"].apply(Age)
```

Employment length

[14]:
```
# Todays date
today = date.today()

# Creating an function to modify the Age Column values.
def EmploymentLength(empLen):
    if empLen > 0:
        return -1

    daysBack = int(empLen)
    joiningDate = (today + timedelta(days= daysBack))
```

ML Model Development

ML development = Turning data into knowledge through feature engineering, model training, and evaluation.

Creating Model

Logistic Regression

[64]:
```
# Creating an instance of the model

lr = LogisticRegression()

# Training the model
lr.fit(x_train_trf, y_train)

# Predicting the values
y_pred = lr.predict(x_test_trf)
```

```
C:\Users\DIWAKAR SINGH\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\linear_mo
to converge (status=1):
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(
```

[65]:
```
print("Accuracy:- ", accuracy_score(y_test, y_pred))
print("Precision:- ", precision_score(y_test, y_pred))
print("Recall:- ", recall_score(y_test, y_pred))
print("F1 Score:- ", f1_score(y_test, y_pred))
print("Confusion Matrix:- ", confusion_matrix(y_test, y_pred))
```

Decision Tree Classifier

[66]:
```
# Creating an instance of the model

dtc = DecisionTreeClassifier(max_depth= 120,
                             criterion= "entropy",
                             min_samples_split= 2,
                             min_samples_leaf= 2)

# Training the model
dtc.fit(x_train_trf, y_train)

# Predicting the values
y_pred = dtc.predict(x_test_trf)
```

[67]:
```
print("Accuracy:- ", accuracy_score(y_test, y_pred))
print("Precision:- ", precision_score(y_test, y_pred))
print("Recall:- ", recall_score(y_test, y_pred))
print("F1 Score:- ", f1_score(y_test, y_pred))
print("Confusion Matrix:- ", confusion_matrix(y_test, y_pred))
print("roc_auc_score:- ", roc_auc_score(y_test, y_pred))
```

```
Accuracy:-  0.9794238683127572
Precision:-  0.22535211267605634
Recall:-  0.14414414414414414
F1 Score:-  0.17582417582417584
Confusion Matrix:-  [[7124   55]
 [  95   16]]
roc_auc_score:-  0.5682414549944846
```

Model Evaluation

Model Evaluation = Assessing how well your model performs on unseen data.

Random Forest Classifier

[68]:
```python
# Creating an instance of the model

rfc = RandomForestClassifier()

# Training the model
rfc.fit(x_train_trf, y_train)

# Predicting the values
y_pred = rfc.predict(x_test_trf)
```

[69]:
```python
print("Accuracy:- ", accuracy_score(y_test, y_pred))
print("Precision:- ", precision_score(y_test, y_pred))
print("Recall:- ", recall_score(y_test, y_pred))
print("F1 Score:- ", f1_score(y_test, y_pred))
print("Confusion Matrix:- ", confusion_matrix(y_test, y_pred))
print("roc_auc_score:- ", roc_auc_score(y_test, y_pred))
```

```
Accuracy:-  0.9849108367626886
Precision:-  0.5217391304347826
Recall:-  0.10810810810810811
F1 Score:-  0.1791044776119403
Confusion Matrix:-  [[7168   11]
 [  99   12]]
roc_auc_score:-  0.5532879306385367
```

AdaBoost Classifier

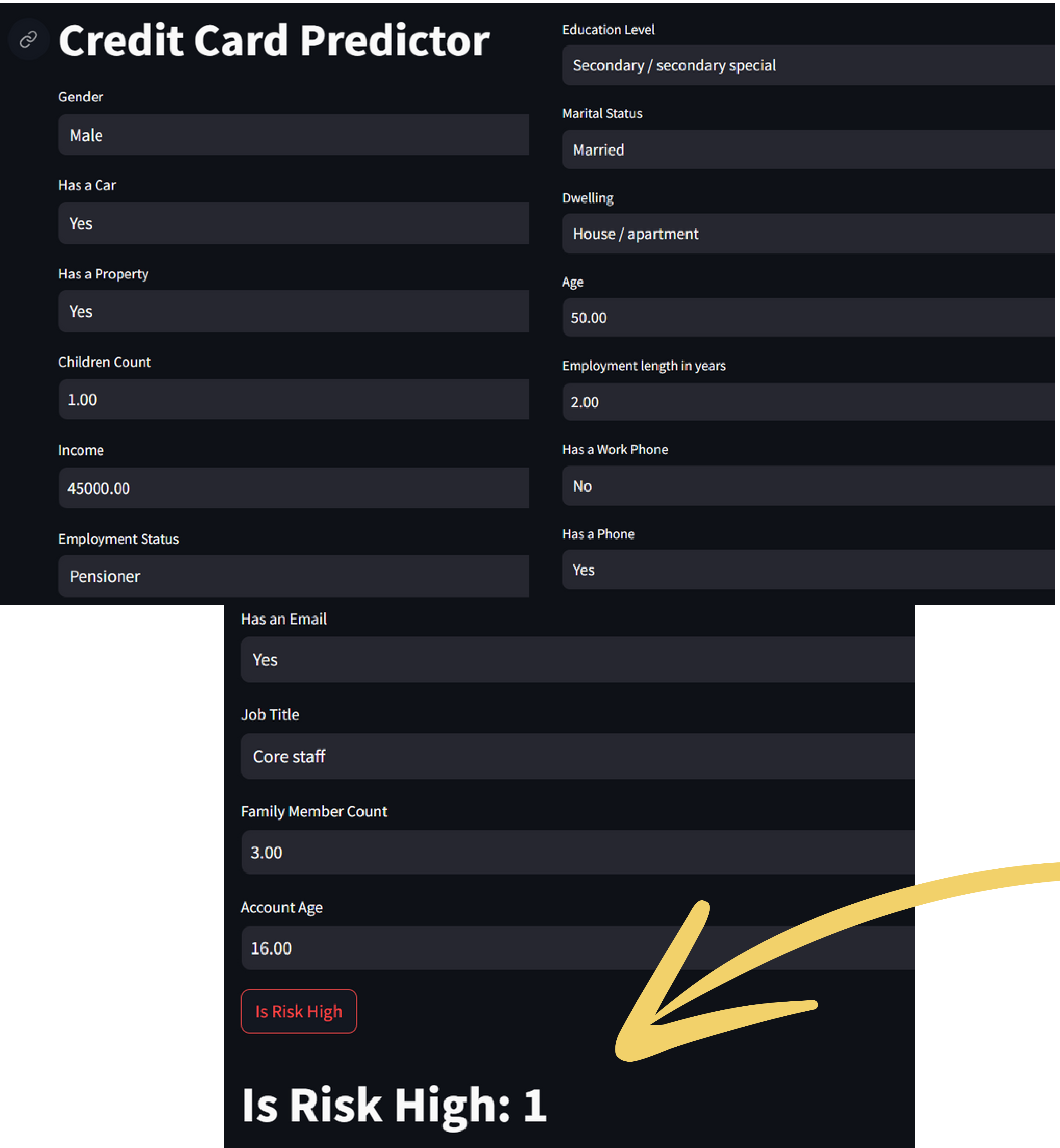
[70]:
```python
# Creating an instance of the model

abc = AdaBoostClassifier()

# Training the model
abc.fit(x_train_trf, y_train)

# Predicting the values
y_pred = abc.predict(x_test_trf)
```

```
C:\Users\DIWAKAR SINGH\AppData\Local\Programs\Python\Python312\Lib\site-
orithm (the default) is deprecated and will be removed in 1.6. Use the S
  warnings.warn(
```

[71]:
```python
print("Accuracy:- ", accuracy_score(y_test, y_pred))
print("Precision:- ", precision_score(y_test, y_pred))
print("Recall:- ", recall_score(y_test, y_pred))
print("F1 Score:- ", f1_score(y_test, y_pred))
print("Confusion Matrix:- ", confusion_matrix(y_test, y_pred))
print("roc_auc_score:- ", roc_auc_score(y_test, y_pred))
```

```
Accuracy:-  0.9847736625514403
Precision:-  0.0
Recall:-  0.0
F1 Score:-  0.0
Confusion Matrix:-  [[7179    0]
 [ 111    0]]
roc_auc_score:-  0.5
```

Predicting Credit Card Risk



Creating a Streamlit, web application to show the model performance and test is using an Interactive UI.

Predicting the Risk, here 1 means the risk is high, and 0 means No Risk.

Recommendations

01

Targeted Monitoring

Focus resources on applications predicted as high-risk (class 1) by **Credit-Card Predictor** model.

02

Tailored Credit Limits

For high-risk but approved applicants, consider offering lower initial credit limits.

03

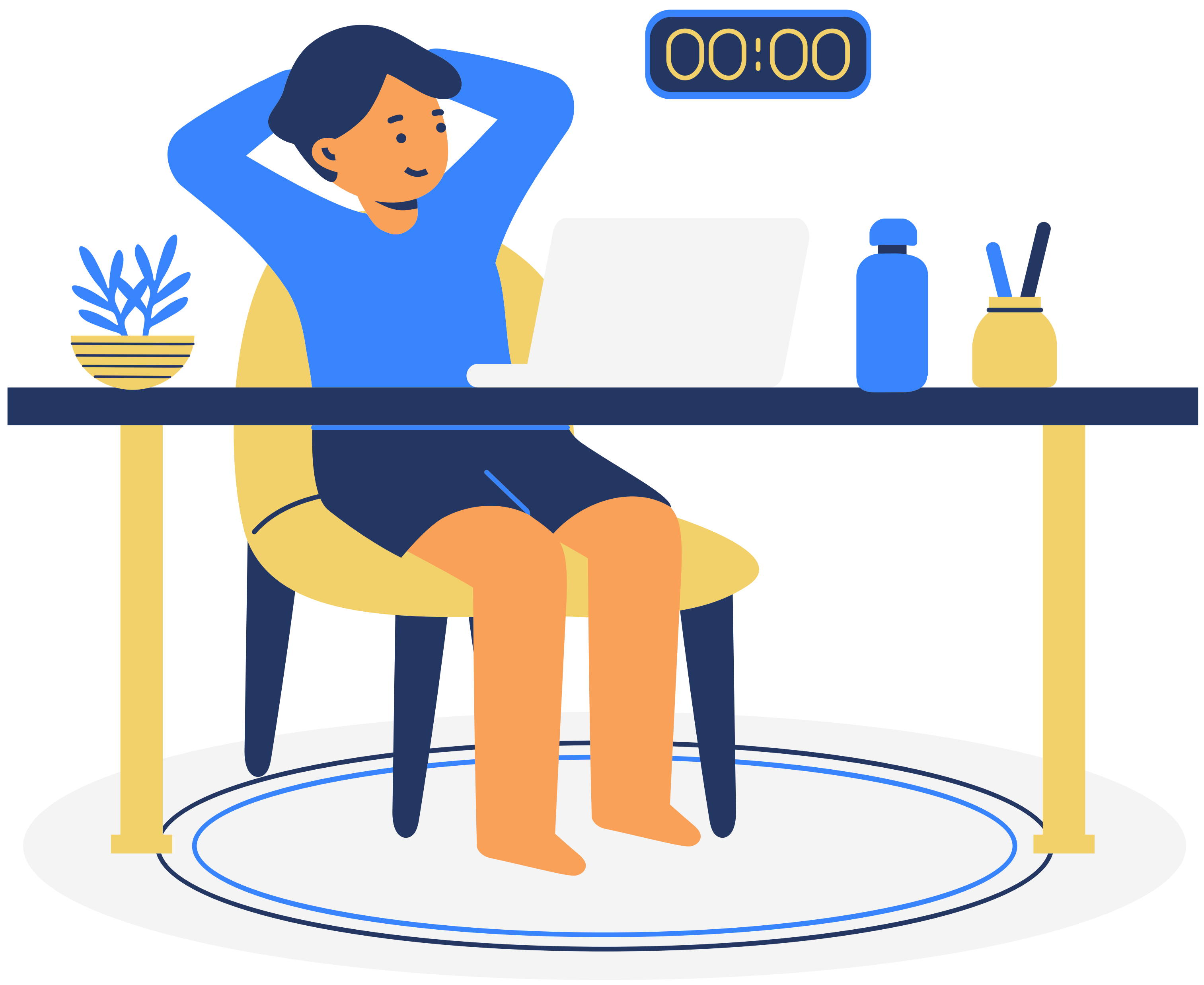
Alternative Products

For high-risk applicants, explore offering alternative credit products with lower limits or secured by collateral to manage risk.

04

Targeted Communication

For applicants predicted as high-risk (class 1), consider proactive outreach with personalized financial advice or credit management resources



Conclusion

By leveraging **Credit Card Model** predictions, we can make informed decisions about credit card applications, potentially reducing risk and improving the overall approval process.

Thank You !!!