Credit Scoring Model

```
from google.colab import drive
# Mount Google Drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
import os
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

```
application_record = pd.read_csv("drive/My Drive/ML_internship/ML_Task01/application_record.csv", index_col=0)
application_record
```

ID	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	NAME_INCOME_TYPE	NAME_EDUCATION_TYPE	NAME_FAMILY_
5008804	M	Y	Y	0	427500.0	Working	Higher education	Civil m
5008805	M	Y	Y	0	427500.0	Working	Higher education	Civil m
5008806	M	Y	Y	0	112500.0	Working	Secondary / secondary special	
5008808	F	N	Y	0	270000.0	Commercial associate	Secondary / secondary special	Single / not
5008809	F	N	Y	0	270000.0	Commercial associate	Secondary / secondary special	Single / not
...	...	...	...	...	...	...	...	
6840104	M	N	Y	0	135000.0	Pensioner	Secondary / secondary special	Se
6840222	F	N	N	0	103500.0	Working	Secondary / secondary special	Single / not
6841878	F	N	N	0	54000.0	Commercial associate	Higher education	Single / not
6842765	F	N	Y	0	72000.0	Pensioner	Secondary / secondary special	
6842885	F	N	Y	0	121500.0	Working	Secondary / secondary special	

438557 rows × 17 columns

```
credit_record = pd.read_csv("drive/My Drive/ML_internship/ML_Task01/credit_record.csv")
credit_record
```

	ID	MONTHS_BALANCE	STATUS
0	5001711	0	X
1	5001711	-1	0
2	5001711	-2	0
3	5001711	-3	0
4	5001712	0	C
...	...	...	...
1048570	5150487	-25	C
1048571	5150487	-26	C
1048572	5150487	-27	C
1048573	5150487	-28	C
1048574	5150487	-29	C

1048575 rows × 3 columns

Cleaning the data

```
# Pivoting the original DataFrame to get STATUS counts as columns
pivoted_df = credit_record.pivot_table(index='ID', columns='STATUS', aggfunc='size', fill_value=0).reset_index()
pivoted_df
```

STATUS	ID	0	1	2	3	4	5	C	X
0	5001711	3	0	0	0	0	0	0	1
1	5001712	10	0	0	0	0	0	9	0
2	5001713	0	0	0	0	0	0	0	22
3	5001714	0	0	0	0	0	0	0	15
4	5001715	0	0	0	0	0	0	0	60
...	...	...	...	...	...	...	...	...	...
45980	5150482	12	0	0	0	0	0	6	0
45981	5150483	0	0	0	0	0	0	0	18
45982	5150484	12	0	0	0	0	0	1	0
45983	5150485	2	0	0	0	0	0	0	0
45984	5150487	0	0	0	0	0	0	30	0

45985 rows × 9 columns

Next steps: Generate code with `pivoted_df` | View recommended plots | New interactive sheet

```
# Conducting InnerJoin between the Application Record and Newly created Pivot Table
merged_df = pd.merge(application_record, pivoted_df, how="inner", on="ID")
merged_df
```

	ID	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	NAME_INCOME_TYPE	NAME_EDUCATION_TYPE	NAME_
0	5008804	M	Y	Y	0	427500.0	Working	Higher education	
1	5008805	M	Y	Y	0	427500.0	Working	Higher education	
2	5008806	M	Y	Y	0	112500.0	Working	Secondary / secondary special	
3	5008808	F	N	Y	0	270000.0	Commercial associate	Secondary / secondary special	Sin
4	5008809	F	N	Y	0	270000.0	Commercial associate	Secondary / secondary special	Sin
...	...	...	...	...	...	...	...	...	
36452	5149828	M	Y	Y	0	315000.0	Working	Secondary / secondary special	
36453	5149834	F	N	Y	0	157500.0	Commercial associate	Higher education	
36454	5149838	F	N	Y	0	157500.0	Pensioner	Higher education	
36455	5150049	F	N	Y	0	283500.0	Working	Secondary / secondary special	
36456	5150337	M	N	Y	0	112500.0	Working	Secondary / secondary special	Sin

36457 rows × 26 columns

```
merged_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 36457 entries, 0 to 36456
Data columns (total 26 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   ID                   36457 non-null  int64
 1   CODE_GENDER          36457 non-null  object
 2   FLAG_OWN_CAR         36457 non-null  object
 3   FLAG_OWN_REALTY      36457 non-null  object
 4   CNT_CHILDREN         36457 non-null  int64
 5   AMT_INCOME_TOTAL     36457 non-null  float64
 6   NAME_INCOME_TYPE     36457 non-null  object
 7   NAME_EDUCATION_TYPE  36457 non-null  object
 8   NAME_FAMILY_STATUS   36457 non-null  object
 9   NAME_HOUSING_TYPE    36457 non-null  object
 10  DAYS_BIRTH           36457 non-null  int64
 11  DAYS_EMPLOYED        36457 non-null  int64
 12  FLAG_MOBIL           36457 non-null  int64
 13  FLAG_WORK_PHONE      36457 non-null  int64
 14  FLAG_PHONE           36457 non-null  int64
 15  FLAG_EMAIL           36457 non-null  int64
 16  OCCUPATION_TYPE      25134 non-null  object
 17  CNT_FAM_MEMBERS      36457 non-null  float64
 18  0                    36457 non-null  int64
 19  1                    36457 non-null  int64
 20  2                    36457 non-null  int64
 21  3                    36457 non-null  int64
 22  4                    36457 non-null  int64
 23  5                    36457 non-null  int64
 24  C                    36457 non-null  int64
 25  X                    36457 non-null  int64
dtypes: float64(2), int64(16), object(8)
memory usage: 7.2+ MB
```

```
# Converting column types to Int 64/Integer
columns = merged_df.iloc[:, 17:].columns
merged_df[columns] = merged_df[columns].astype('Int64')
```

```
merged_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 36457 entries, 0 to 36456
Data columns (total 26 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   ID                   36457 non-null  int64
```

```
 1   CODE_GENDER          36457 non-null  object
 2   FLAG_OWN_CAR         36457 non-null  object
 3   FLAG_OWN_REALTY      36457 non-null  object
 4   CNT_CHILDREN         36457 non-null  int64
 5   AMT_INCOME_TOTAL     36457 non-null  float64
 6   NAME_INCOME_TYPE     36457 non-null  object
 7   NAME_EDUCATION_TYPE  36457 non-null  object
 8   NAME_FAMILY_STATUS   36457 non-null  object
 9   NAME_HOUSING_TYPE    36457 non-null  object
 10  DAYS_BIRTH           36457 non-null  int64
 11  DAYS_EMPLOYED        36457 non-null  int64
 12  FLAG_MOBIL           36457 non-null  int64
 13  FLAG_WORK_PHONE      36457 non-null  int64
 14  FLAG_PHONE           36457 non-null  int64
 15  FLAG_EMAIL           36457 non-null  int64
 16  OCCUPATION_TYPE      25134 non-null  object
 17  CNT_FAM_MEMBERS      36457 non-null  Int64
 18  0                    36457 non-null  Int64
 19  1                    36457 non-null  Int64
 20  2                    36457 non-null  Int64
 21  3                    36457 non-null  Int64
 22  4                    36457 non-null  Int64
 23  5                    36457 non-null  Int64
 24  C                    36457 non-null  Int64
 25  X                    36457 non-null  Int64
dtypes: Int64(9), float64(1), int64(8), object(8)
memory usage: 7.5+ MB
```

```
merged_df.describe()
```

	ID	CNT_CHILDREN	AMT_INCOME_TOTAL	DAYS_BIRTH	DAYS_EMPLOYED	FLAG_MOBIL	FLAG_WORK_PHONE	FLAG_PHONE	FLAG_EMAIL
count	3.645700e+04	36457.000000	3.645700e+04	36457.000000	36457.000000	36457.0	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
std	4.187524e+04	0.742367	1.017892e+05	4200.549944	137651.334859	0.0	0.417934	0.455965	0.285787
min	5.008804e+06	0.000000	2.700000e+04	-25152.000000	-15713.000000	1.0	0.000000	0.000000	0.000000
25%	5.042028e+06	0.000000	1.215000e+05	-19438.000000	-3153.000000	1.0	0.000000	0.000000	0.000000
50%	5.074614e+06	0.000000	1.575000e+05	-15563.000000	-1552.000000	1.0	0.000000	0.000000	0.000000
75%	5.115396e+06	1.000000	2.250000e+05	-12462.000000	-408.000000	1.0	0.000000	1.000000	0.000000
max	5.150487e+06	19.000000	1.575000e+06	-7489.000000	365243.000000	1.0	1.000000	1.000000	1.000000

```
merged_df['DAYS_EMPLOYED'].sort_values(ascending = False).unique()
```

```
array([365243,    -17,    -43, ..., -15227, -15661, -15713])
```

```
# Positive values
merged_df['DAYS_EMPLOYED'] = merged_df['DAYS_EMPLOYED'].apply(lambda x : 0 if x > 0 else -x)
merged_df['DAYS_BIRTH'] = merged_df['DAYS_BIRTH'] * -1

# Renaming columns
merged_df.rename(columns={'CODE_GENDER':'Gender','FLAG_OWN_CAR':'Car','FLAG_OWN_REALTY':'Property','CNT_CHILDREN':'Children','AMT_INCOME_TOT
'NAME_INCOME_TYPE':'Income status', 'NAME_EDUCATION_TYPE':'Education','NAME_FAMILY_STATUS':'Marital', 'NAME_HOUSING_TYPE':'Housing', 'DAYS_B
'DAYS_EMPLOYED' : 'Days employed', 'FLAG_MOBIL' : 'Mobile', 'FLAG_WORK_PHONE':'Work phone', 'FLAG_PHONE':'Phone', 'FLAG_EMAIL':'Email', 'OCC
'CNT_FAM_MEMBERS':'Family'},inplace=True)
```

```
merged_df.head()
```

	ID	Gender	Car	Property	Children	Income	Income status	Education	Marital	Housing	...	Job	Family	0	1	2	3	4	5	C
0	5008804	M	Y	Y	0	427500.0	Working	Higher education	Civil marriage	Rented apartment	...	NaN	2	1	1	0	0	0	0	13
1	5008805	M	Y	Y	0	427500.0	Working	Higher education	Civil marriage	Rented apartment	...	NaN	2	1	1	0	0	0	0	12
2	5008806	M	Y	Y	0	112500.0	Working	Secondary / secondary special	Married	House / apartment	...	Security staff	2	7	0	0	0	0	0	7
3	5008808	F	N	Y	0	270000.0	Commercial associate	Secondary / secondary special	Single / not married	House / apartment	...	Sales staff	1	2	0	0	0	0	0	0
4	5008809	F	N	Y	0	270000.0	Commercial associate	Secondary / secondary special	Single / not married	House / apartment	...	Sales staff	1	0	0	0	0	0	0	0

5 rows × 26 columns

```
merged_df.Mobile.unique()
```

```
array([1])
```

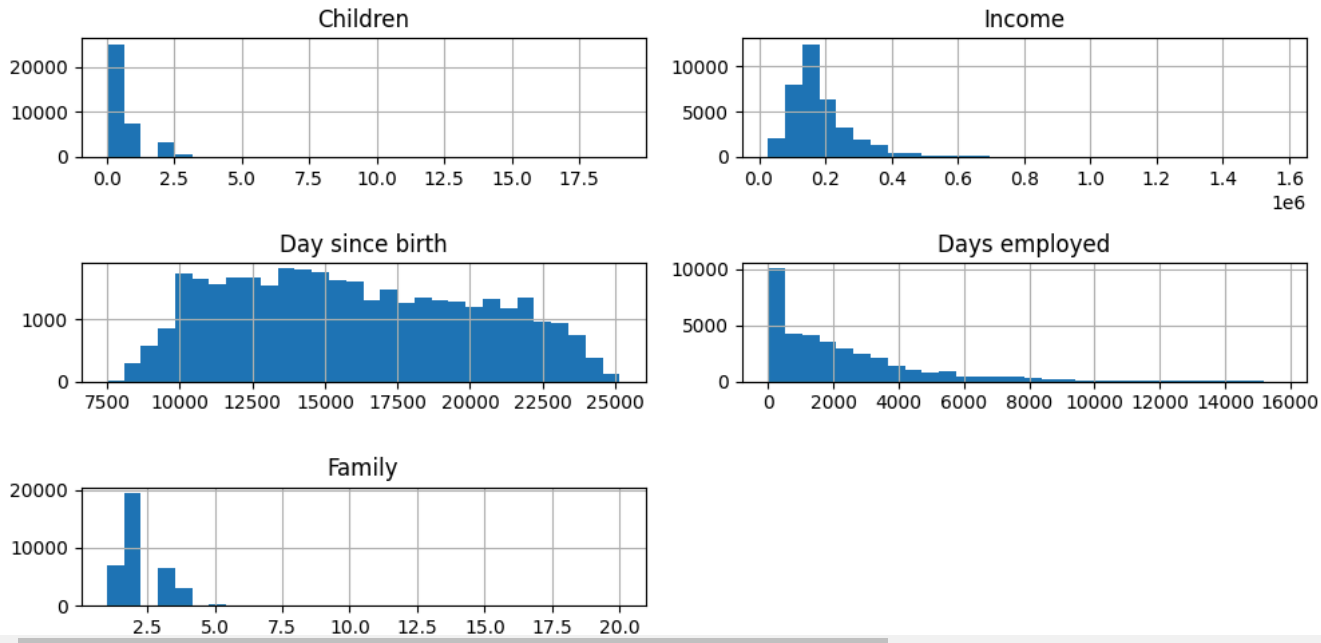
```
merged_df = merged_df.drop(['ID', 'Mobile'], axis = 1)
```

```
import math
# Summing the column from 1 to 5
merged_df['Bad debt chance'] = merged_df.iloc[:,17:-2].sum(axis=1, min_count=1)
# Low if the sum is 0, else high
merged_df['Bad debt chance'] = merged_df['Bad debt chance'].apply(lambda x: 'low' if (x == 0) else 'high')
merged_df['Bad debt chance'].value_counts()
```

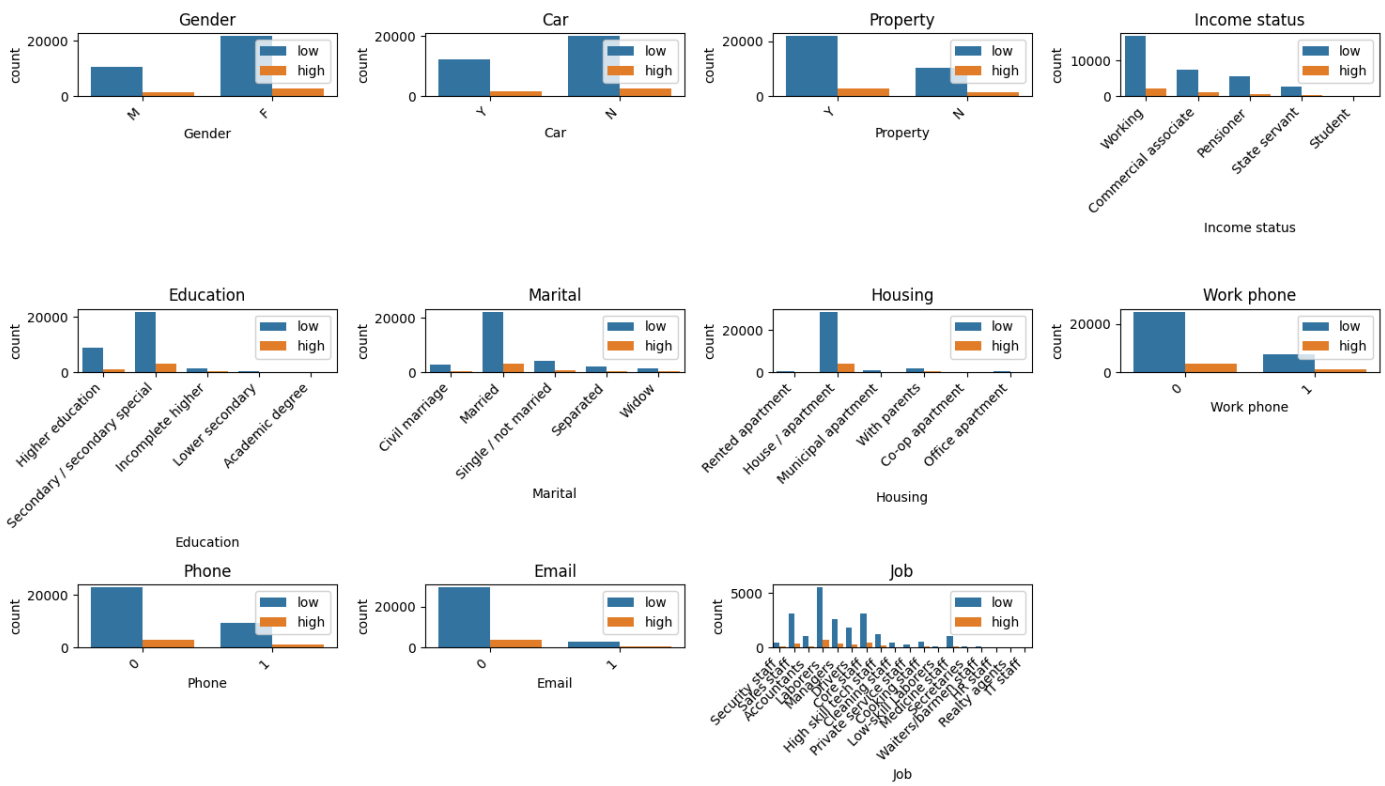
Bad debt chance	count
low	32166
high	4291

```
# Drop the status columns since we summarized it with bad debt chance
merged_df = merged_df.drop(merged_df.iloc[:,16:-1].columns, axis = 1)
```

```
# Histogram with numerical features
merged_df.drop(['Work phone', 'Phone', 'Email'], axis = 1).hist(bins=30, figsize=(10, 5))
plt.tight_layout()
```



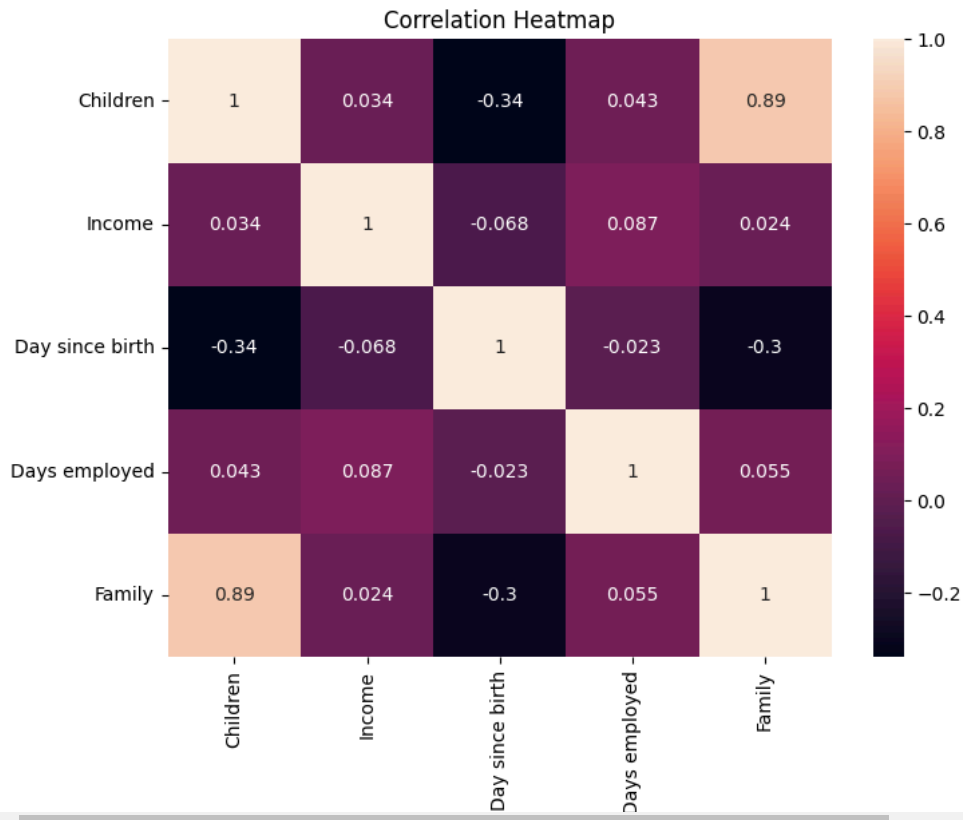

```
# Countplot with categorical features
features = ['Gender', 'Car', 'Property', 'Income status', 'Education', 'Marital', 'Housing', 'Work phone', 'Phone', 'Email', 'Job']
plt.figure(figsize=(15, 10))

for feature in features:
    plt.subplot(4, 4, features.index(feature) + 1)
    sns.countplot(data=merged_df, x=feature, hue = 'Bad debt chance', hue_order=['low', 'high'])
    plt.xticks(rotation=45, ha='right')
    plt.title(feature)
    plt.legend(loc='best')

plt.tight_layout()
plt.show()
```



Correlation heatmap

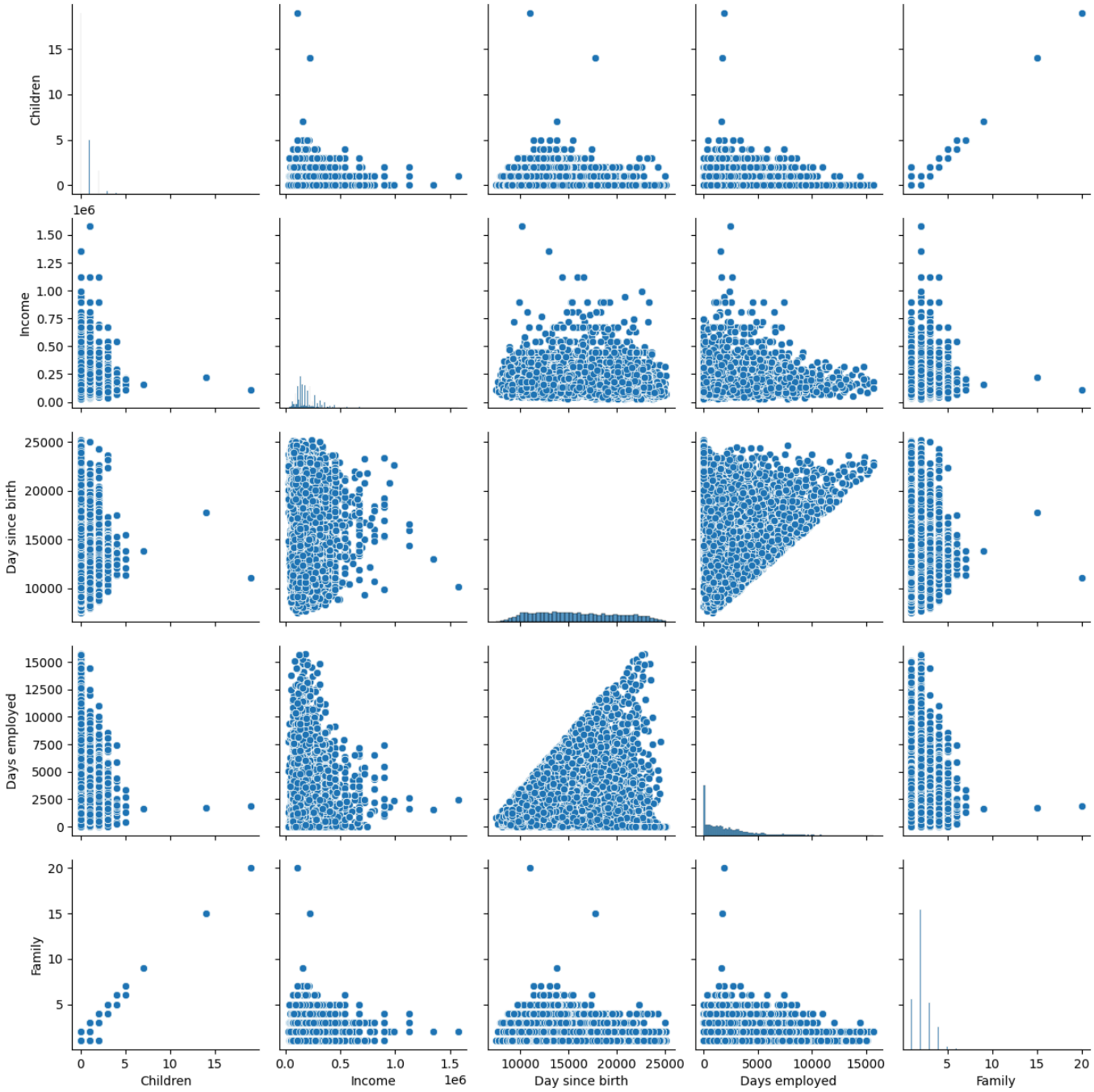
```
plt.figure(figsize = (8, 6))
sns.heatmap(merged_df.select_dtypes(include =['float64', 'int64']).drop(['Email', 'Phone', 'Work phone'], axis = 1).corr(), annot = True)
plt.title('Correlation Heatmap')
```

```
Text(0.5, 1.0, 'Correlation Heatmap')
```




```
sns.pairplot(merged_df.select_dtypes(include =['float64', 'int64']).drop(['Email', 'Phone', 'Work phone'], axis = 1))
```

```
<seaborn.axisgrid.PairGrid at 0x7b94350a9c30>
```



Data Preprocessing

```
# Engineering income category for stratified sample
def cat(x):
    if x <= 100000:
        return "low"
    elif x <= 300000:
        return 'medium'
    else:
        return 'high'

merged_df['Income_cat'] = merged_df['Income'].apply(cat)
merged_df['Income_cat'].value_counts()
```

Income_cat	count
medium	27542
low	5086
high	3829

Feature selection

Double-click (or enter) to edit

```
# Defining stratified train and test set
from sklearn.model_selection import train_test_split

X = merged_df.drop(['Family', 'Bad debt chance'], axis = 1)
y = merged_df['Bad debt chance']


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42, stratify=X['Income_cat'])


X_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 25519 entries, 34239 to 24918
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Gender           25519 non-null  object
 1   Car              25519 non-null  object
 2   Property         25519 non-null  object
 3   Children         25519 non-null  int64
 4   Income           25519 non-null  float64
 5   Income status    25519 non-null  object
 6   Education        25519 non-null  object
 7   Marital          25519 non-null  object
 8   Housing          25519 non-null  object
 9   Day since birth  25519 non-null  int64
 10  Days employed    25519 non-null  int64
 11  Work phone       25519 non-null  int64
 12  Phone            25519 non-null  int64
 13  Email            25519 non-null  int64
 14  Job              17550 non-null  object
 15  Income_cat       25519 non-null  object
dtypes: float64(1), int64(6), object(9)
memory usage: 3.3+ MB
```

Creating a Data Pipeline for Classification: preprocessing Numerical and categorical Features

```
from sklearn.pipeline import Pipeline, FeatureUnion
from sklearn.base import BaseEstimator, TransformerMixin
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.model_selection import cross_val_score

class DataFrameSelector(BaseEstimator, TransformerMixin):
    def __init__(self, attribute_names):
        self.attribute_names = attribute_names
    def fit(self, X, y=None):
        return self
    def transform(self, X):
        return X[self.attribute_names].values

cat_attribs = X_train.drop(['Children', 'Income', 'Day since birth', 'Days employed', 'Income_cat'], axis = 1).column
num_attribs = X_train.iloc[:,[3, 4, 9, 10]].columns

cat_pipeline = Pipeline([
    ('selector', DataFrameSelector(cat_attribs)),
    ('one_hot_encoder', OneHotEncoder())
])
```

```
num_pipeline = Pipeline([
    ('selector', DataFrameSelector(num_attribs)),
    ('imputer', SimpleImputer(strategy='mean')),
    ('std_scaler', StandardScaler())
])

full_pipeline = FeatureUnion(transformer_list=[
    ('cat_pipeline', cat_pipeline),
    ('num_pipeline', num_pipeline),
])


# Transform the training and testing data sets
X_train_prepared = full_pipeline.fit_transform(X_train)
X_test_prepared = full_pipeline.fit_transform(X_test)
```

Implementing Logistic Regression

```
# Logistic Regression
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, classification_report

model = LogisticRegression(solver = 'liblinear', C=1e9, class_weight = {'low':2,'high':8},random_state = 0)
model.fit(X_train_prepared, y_train)
```

```
LogisticRegression
LogisticRegression(C=1000000000.0, class_weight={'high': 8, 'low': 2},
                   random_state=0, solver='liblinear')
```