Data Cleaning, EDA, and Feature Engineering Notebook

Contents

Importing Packages

In [1]:
```
# Importing Packages
import numpy as np
import pandas as pd
import re
import json
import requests
import warnings
warnings.filterwarnings('ignore')
pd.set_option("display.max_rows", 999)
pd.set_option("display.max_columns", 999)

import matplotlib.pyplot as plt
from matplotlib import style
import seaborn as sns
%matplotlib inline
plt.style.use("fivethirtyeight")

from sklearn.model_selection import train_test_split
import pickle
```

Uploading Data

In [2]:
```
df = pd.read_excel("../data/default of credit card clients.xls")
new_header = df.iloc[0]
df = df[1:]
df.columns = new_header
df = df.rename(columns={"default payment next month": "default"})
df.head()
```

Out[2]:

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	PAY_5	PAY_6	BILL_AMT1	BILL_AMT2	BILL_AMT3	BILL_A
1	1	20000	2	2	1	24	2	2	-1	-1	-2	-2	3913	3102	689	
2	2	120000	2	2	2	26	-1	2	0	0	0	2	2682	1725	2682	3
3	3	90000	2	2	2	34	0	0	0	0	0	0	29239	14027	13559	1
4	4	50000	2	2	1	37	0	0	0	0	0	0	46990	48233	49291	2
5	5	50000	1	2	1	57	-1	0	-1	0	0	0	8617	5670	35835	2

In [3]:
```
df.dtypes
```

Out[3]:
```
0
ID           object
LIMIT_BAL    object
SEX          object
EDUCATION    object
MARRIAGE     object
AGE          object
PAY_0        object
PAY_2        object
PAY_3        object
PAY_4        object
PAY_5        object
PAY_6        object
BILL_AMT1    object
BILL_AMT2    object
BILL_AMT3    object
```

```
BILL_AMT4    object
BILL_AMT5    object
BILL_AMT6    object
PAY_AMT1     object
PAY_AMT2     object
PAY_AMT3     object
PAY_AMT4     object
PAY_AMT5     object
PAY_AMT6     object
default      object
dtype: object
```

Creating Train, Validation, and Testing Sets

In [4]:
```
X = df.drop(["default"], axis=1)
y = df["default"]
# split the full data 80:20 into training:validation sets
X_train, X_val, y_train, y_val = train_test_split(X, y, train_size=0.8, random_state=42)
# split training data 87.5:12.5 into training:testing sets
X_tr, X_tt, y_tr, y_tt = train_test_split(X_train, y_train, train_size=0.875, random_state=42)
train = pd.concat([X_tr, y_tr], axis=1)
val = pd.concat([X_val, y_val], axis=1)
```

In [5]:
```
# X_tt.to_csv("../data/testing.csv")
# y_tt.to_csv("../data/testing_labels.csv")
# train.to_csv("../data/training.csv", index=False)
# val.to_csv("../data/validate.csv", index=False)
```

Data Cleaning

In [6]:
```
tr = pd.read_csv("../data/training.csv")
val = pd.read_csv("../data/validate.csv")
tr = tr.drop(["ID"], axis=1)
val = val.drop(["ID"], axis=1)
tr.dtypes
```

Out[6]:
```
LIMIT_BAL    int64
SEX          int64
EDUCATION    int64
MARRIAGE     int64
AGE          int64
PAY_0        int64
PAY_2        int64
PAY_3        int64
PAY_4        int64
PAY_5        int64
PAY_6        int64
BILL_AMT1    int64
BILL_AMT2    int64
BILL_AMT3    int64
BILL_AMT4    int64
BILL_AMT5    int64
BILL_AMT6    int64
PAY_AMT1     int64
PAY_AMT2     int64
PAY_AMT3     int64
PAY_AMT4     int64
PAY_AMT5     int64
PAY_AMT6     int64
default      int64
dtype: object
```

In [7]:
```
tr.head()
```

Out[7]:

	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	PAY_5	PAY_6	BILL_AMT1	BILL_AMT2	BILL_AMT3	BILL_AMT4
0	50000	2	2	1	44	0	0	0	0	0	0	45578	41906	35703	22360
1	160000	2	3	1	46	-1	-1	-1	0	-1	-1	24904	2338	4856	4127
2	100000	2	2	1	47	-1	-1	-1	-1	-1	-2	6666	6666	0	6270
3	170000	2	2	1	29	0	0	0	0	0	0	79091	62575	63317	63903
4	150000	2	1	2	33	-2	-2	-2	-2	-2	-2	24393	26847	32702	33459

In [8]:
```
# look for anomalies in minimum and maximum values and peculiarities in interquartile values
tr.describe()
```

Out[8]:

	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	
count	21000.000000	21000.000000	21000.000000	21000.000000	21000.000000	21000.000000	21000.000000	21000.000000	21000.000000	210
mean	167214.937143	1.605190	1.854333	1.549476	35.538286	-0.015524	-0.130952	-0.165667	-0.216952	
std	129561.159854	0.488821	0.794086	0.523126	9.257936	1.127750	1.200066	1.194393	1.170611	
min	10000.000000	1.000000	0.000000	0.000000	21.000000	-2.000000	-2.000000	-2.000000	-2.000000	
25%	50000.000000	1.000000	1.000000	1.000000	28.000000	-1.000000	-1.000000	-1.000000	-1.000000	
50%	140000.000000	2.000000	2.000000	2.000000	34.000000	0.000000	0.000000	0.000000	0.000000	
75%	240000.000000	2.000000	2.000000	2.000000	42.000000	0.000000	0.000000	0.000000	0.000000	
max	1000000.000000	2.000000	6.000000	3.000000	79.000000	8.000000	8.000000	8.000000	8.000000	

In [9]:
```
# Use requests and json modules to webscrape current exchange rate for TWD to USD
url = 'https://openexchangerates.org/api/latest.json?app_id=c51b1508fb4145259b1c2fade72a2c04'
response = requests.get(url)
data = response.json()
rate = data['rates']['TWD']
```

In [10]:
```
# check for null values
data = [tr, val]
for d in data:
    print(d.isna().sum())
```

```
LIMIT_BAL    0
SEX          0
EDUCATION    0
MARRIAGE     0
AGE          0
PAY_0        0
PAY_2        0
PAY_3        0
PAY_4        0
PAY_5        0
PAY_6        0
BILL_AMT1    0
BILL_AMT2    0
BILL_AMT3    0
BILL_AMT4    0
BILL_AMT5    0
BILL_AMT6    0
PAY_AMT1     0
PAY_AMT2     0
PAY_AMT3     0
PAY_AMT4     0
PAY_AMT5     0
PAY_AMT6     0
default      0
dtype: int64
LIMIT_BAL    0
SEX          0
EDUCATION    0
MARRIAGE     0
AGE          0
PAY_0        0
PAY_2        0
PAY_3        0
PAY_4        0
PAY_5        0
PAY_6        0
BILL_AMT1    0
BILL_AMT2    0
BILL_AMT3    0
BILL_AMT4    0
BILL_AMT5    0
BILL_AMT6    0
PAY_AMT1     0
PAY_AMT2     0
PAY_AMT3     0
PAY_AMT4     0
PAY_AMT5     0
PAY_AMT6     0
default      0
dtype: int64
```

In [11]:
```
# change column names for easier reference
for d in data:
    d.rename(columns={"PAY_0": "behind1",
                      "PAY_2": "behind2",
                      "PAY_3": "behind3",
                      "PAY_4": "behind4",
                      "PAY_5": "behind5",
                      "PAY_6": "behind6",
```

```
                          "BILL_AMT1": "billed1",
                          "BILL_AMT2": "billed2",
                          "BILL_AMT3": "billed3",
                          "BILL_AMT4": "billed4",
                          "BILL_AMT5": "billed5",
                          "BILL_AMT6": "billed6",
                          "PAY_AMT1": "paid1",
                          "PAY_AMT2": "paid2",
                          "PAY_AMT3": "paid3",
                          "PAY_AMT4": "paid4",
                          "PAY_AMT5": "paid5",
                          "PAY_AMT6": "paid6",
                          "SEX": "gender",
                          "EDUCATION": "education",
                          "MARRIAGE": "marriage",
                          "AGE": "age",
                          "LIMIT_BAL": "limit"}, inplace=True)
```

In [12]:
```
# Change all Taiwanese to US Dollars for better understanding for American audience
for d in data:
    d[['limit']] = d[['limit']]/rate
    d[['billed1', 'billed2', 'billed3', 'billed4', 'billed5', 'billed6']] = d[['billed1', 'billed2', 'billed3', 'billed4'
    d[['paid1', 'paid2', 'paid3', 'paid4', 'paid5', 'paid6']] = d[['paid1', 'paid2', 'paid3', 'paid4', 'paid5', 'paid6']]
```

In [13]:
```
# Round all decimals to two decimal places to represent cents
for d in data:
    d['limit'] = d['limit'].apply(lambda x: round(x, 2))
    d[['billed1', 'billed2', 'billed3', 'billed4', 'billed5', 'billed6']] = d[['billed1', 'billed2', 'billed3', 'billed4'
    d[['paid1', 'paid2', 'paid3', 'paid4', 'paid5', 'paid6']] = d[['paid1', 'paid2', 'paid3', 'paid4', 'paid5', 'paid6']]
```

In [14]:
```
# put all zero values into category of 3 (other) for marriage
# lump all the other and unknown education categories together:  0, 5, 6 values to category 4 (other)
for d in data:
    d.replace({'marriage': {0:3}}, inplace=True)
    d.replace({'education': {5:4, 0:4, 6:4}}, inplace=True)
```

Pickle out baseline models

In [15]:
```
tr.head()
```

Out[15]:

	limit	gender	education	marriage	age	behind1	behind2	behind3	behind4	behind5	behind6	billed1	billed2	billed3	billed4	bille
0	1791.02	2	2	1	44	0	0	0	0	0	0	1632.63	1501.09	1278.90	800.95	847.
1	5731.27	2	3	1	46	-1	-1	-1	0	-1	-1	892.07	83.75	173.94	147.83	143
2	3582.05	2	2	1	47	-1	-1	-1	-1	-1	-2	238.78	238.78	0.00	224.59	-14
3	6089.48	2	2	1	29	0	0	0	0	0	0	2833.08	2241.47	2268.04	2289.04	1558
4	5373.07	2	1	2	33	-2	-2	-2	-2	-2	-2	873.77	961.67	1171.40	1198.52	995

In [16]:
```
# pickle_out = open("../data/pickles/training_cleaned.pickle","wb")
# pickle.dump(tr, pickle_out)
# pickle_out.close()
```

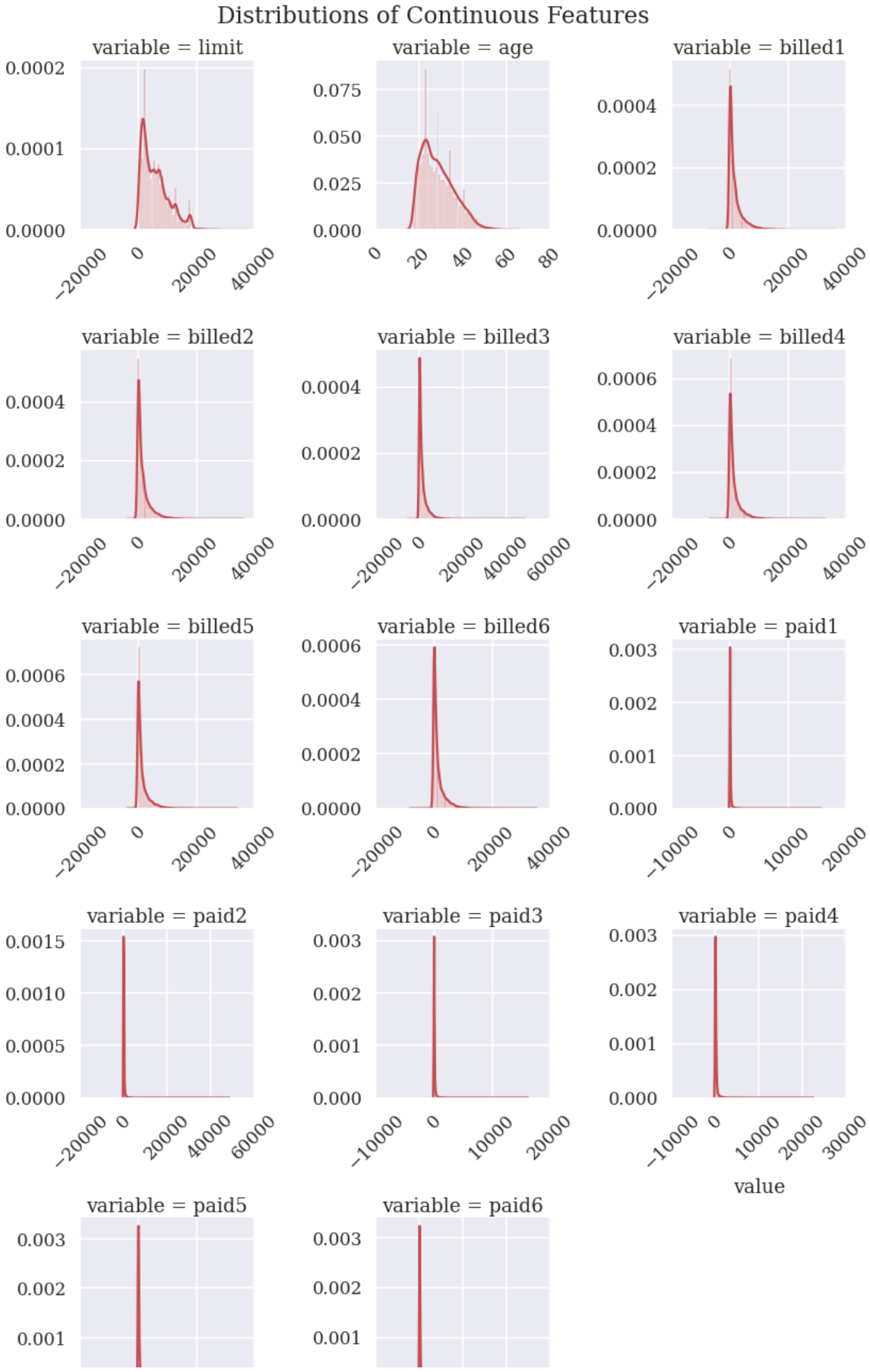
In [17]:
```
# pickle_out = open("../data/pickles/validate_cleaned.pickle","wb")
# pickle.dump(val, pickle_out)
# pickle_out.close()
```

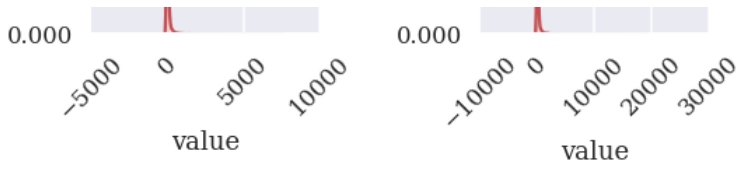
Exploratory Data Analysis

In [18]:
```
# organize features into categorical and continuous
categorical = tr[['gender', 'marriage', 'education', 'behind1', 'behind2', 'behind3', 'behind4', 'behind5', 'behind6']]
continuous = tr[['limit', 'age', 'billed1', 'billed2', 'billed3', 'billed4', 'billed5', 'billed6', 'paid1', 'paid2', 'pai
cat_col = categorical.columns
cont_col = continuous.columns
```

In [19]:
```
# display distributions of all the continuous variables

con_1 = pd.melt(tr, value_vars = cont_col)
sns.set_theme(style="darkgrid", font='serif', context='talk')
g = sns.FacetGrid(con_1, col='variable', col_wrap=3, sharex=False, sharey=False, height=4)
```

```
g = g.map(sns.distplot, 'value', color='r')
g.set_xticklabels(rotation=45)
g.fig.subplots_adjust(top=0.9)
g.fig.suptitle("Distributions of Continuous Features")
g.fig.tight_layout()
# plt.savefig("../images/distplot.png")
```

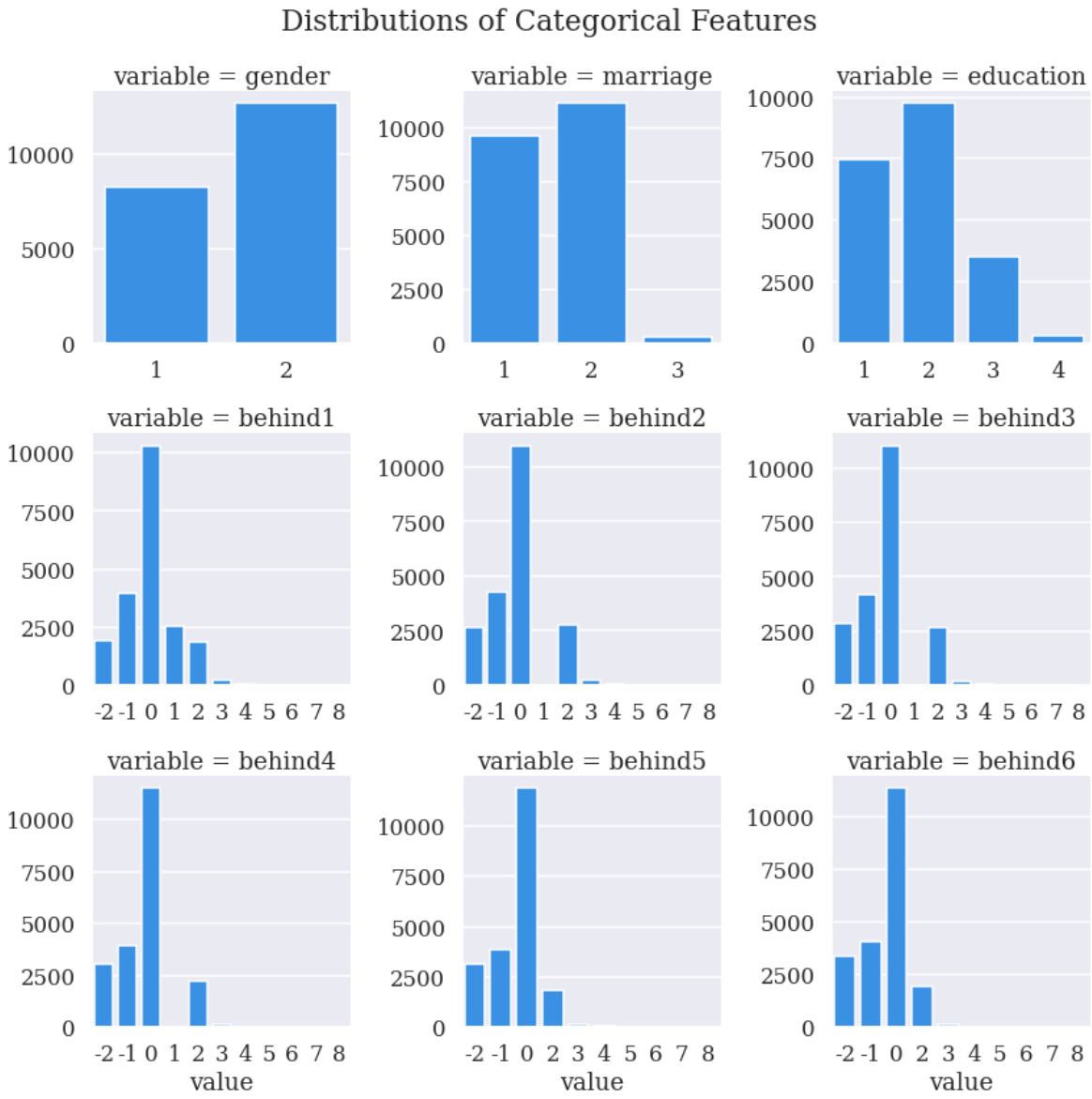




-img src="../images/distplot.png">

In [20]:

```
# Use bar graphs of the distribution of data for categorical variables

cat_1 = pd.melt(tr, value_vars=cat_col)
sns.set_theme(style="darkgrid", font='serif', context='talk')
g = sns.FacetGrid(cat_1, col='variable', col_wrap=3, sharex=False, sharey=False, height=4)
g = g.map(sns.countplot, 'value', color='dodgerblue')
g.set_xticklabels()
g.fig.subplots_adjust(top=0.9)
g.fig.suptitle("Distributions of Categorical Features")
g.fig.tight_layout()
# plt.savefig("../images/countplot.png")
```

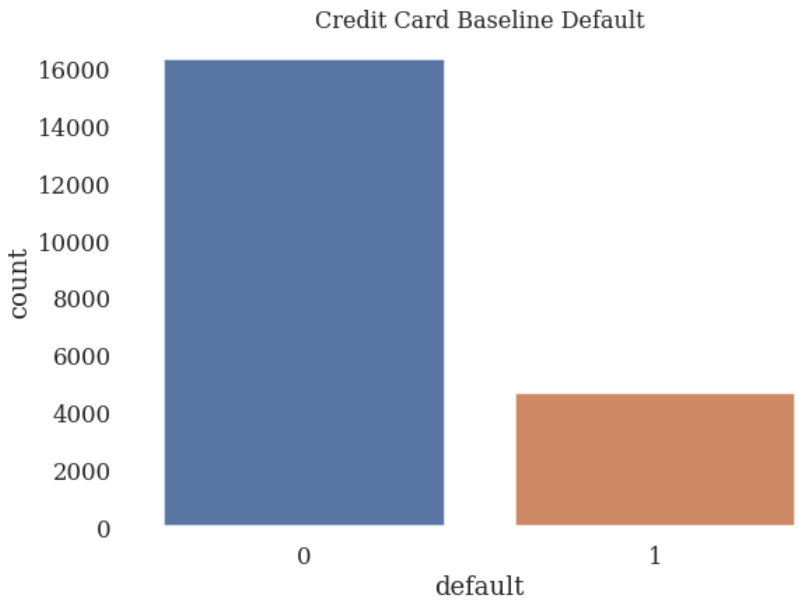


img src="../images/countplot.png">

In [21]:

```
yes = tr.default.sum()
no = len(tr)-yes
perc_y = round(yes/len(tr)*100, 1)
perc_n = round(no/len(tr)*100, 1)

plt.figure(figsize=(8,6))
sns.set_theme(style="darkgrid", font='serif', context='talk')
sns.countplot('default', data=tr)
```

```
plt.title('Credit Card Baseline Default', size=16)
plt.box(False);
# plt.savefig("../images/baseline.png")
```



img src="../images/baseline.png">

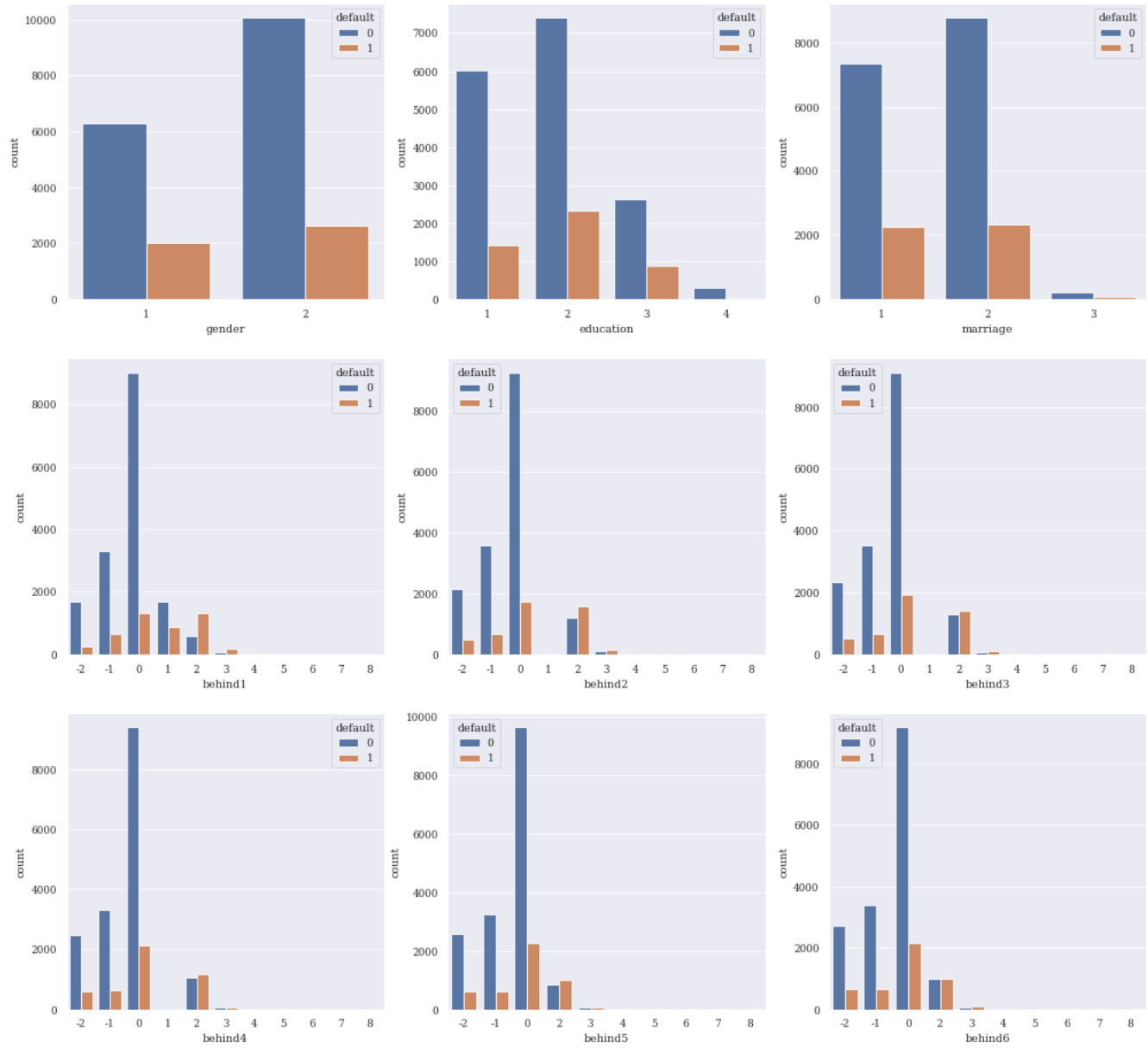
In [22]:
```
print("Number of Total Non-Defaulters: ", yes)
print("Number of Defaulters: ", no)
print("Percentage of Non-Defaulters: ", perc_y)
print("Percentage of Defaulters: ", perc_n)

pd.DataFrame
default = pd.DataFrame(data = {"Training Dataset": [yes, no, perc_y, perc_n]},
                       index = ["Number of Total Non-Defaulters: ", "Number of Defaulters: ", "Percentage of Non-Defaulte
default
```

```
Number of Total Non-Defaulters:  4656
Number of Defaulters:  16344
Percentage of Non-Defaulters:  22.2
Percentage of Defaulters:  77.8
```

Out[22]:

	Training Dataset
Number of Total Non-Defaulters:	4656.0
Number of Defaulters:	16344.0
Percentage of Non-Defaulters:	22.2
Percentage of Defaulters:	77.8

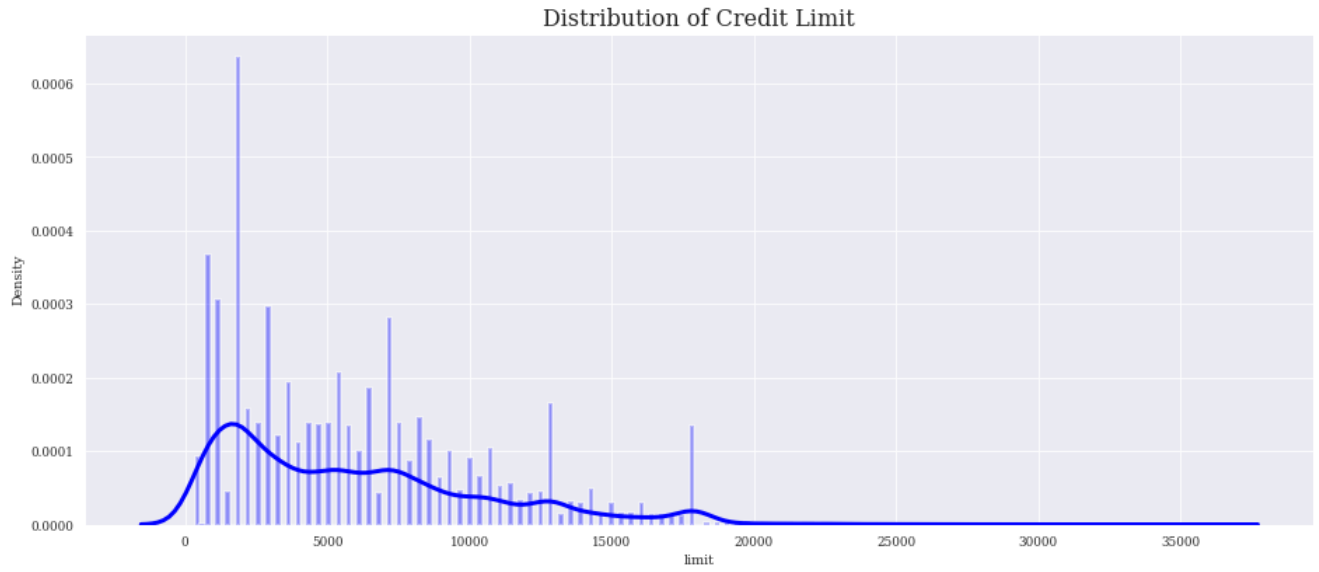
In [28]:
```
subset = tr[['gender', 'education', 'marriage', 'behind1', 'behind2', 'behind3', 'behind4', 'behind5', 'behind6', 'defaul
f, axes = plt.subplots(3, 3, figsize=(15, 15), facecolor='white')
sns.set_theme(style="darkgrid", font='serif', context='paper')
f.suptitle('Frequency of Categorical Variables', size=16)
ax1 = sns.countplot(x="gender", hue="default", data=subset, ax=axes[0,0])
ax2 = sns.countplot(x="education", hue="default", data=subset, ax=axes[0,1])
ax3 = sns.countplot(x="marriage", hue="default", data=subset, ax=axes[0,2])
ax4 = sns.countplot(x="behind1", hue="default", data=subset, ax=axes[1,0])
ax5 = sns.countplot(x="behind2", hue="default", data=subset, ax=axes[1,1])
ax6 = sns.countplot(x="behind3", hue="default", data=subset, ax=axes[1,2])
ax7 = sns.countplot(x="behind4", hue="default", data=subset, ax=axes[2,0])
ax8 = sns.countplot(x="behind5", hue="default", data=subset, ax=axes[2,1])
ax9 = sns.countplot(x="behind6", hue="default", data=subset, ax=axes[2,2])
# plt.savefig("../images/default_freq_by_cat.png")
```



-img src="../images/default_freq_by_cat.png">

In [29]:

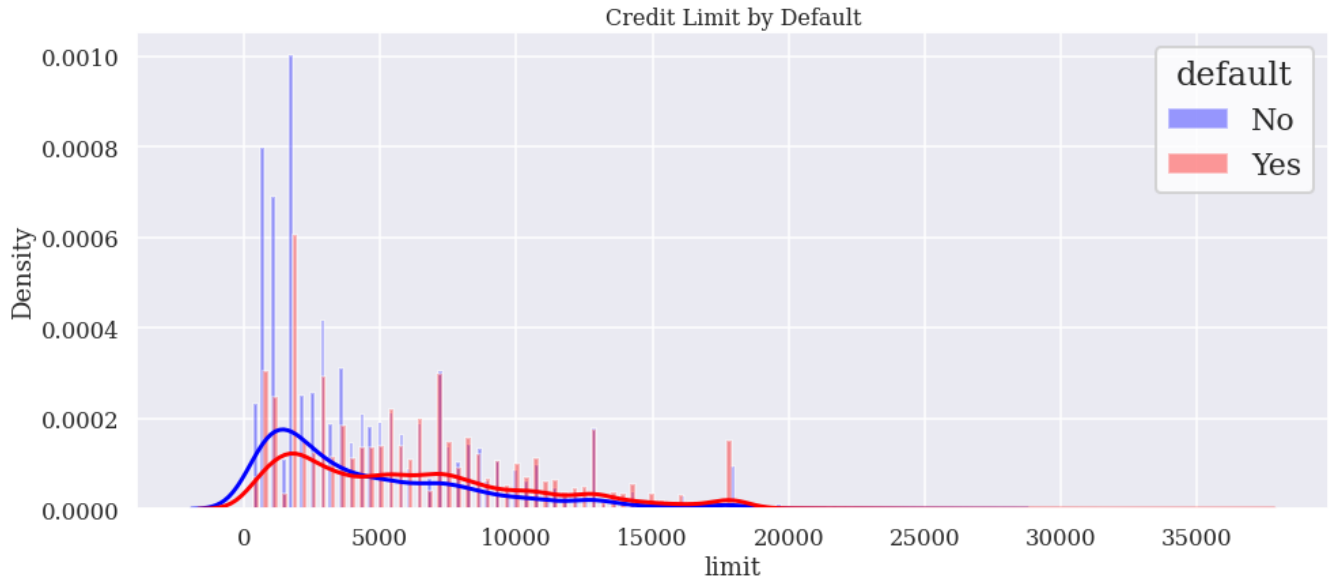
```
plt.figure(figsize = (14,6))
plt.title('Distribution of Credit Limit', size=16)
sns.set_theme(style="darkgrid", font='serif', context='poster')
g = sns.distplot(tr['limit'], kde=True, bins=200, color="blue")
plt.show()
# plt.savefig("../images/credit_limit.png")
```



-img src="../images/credit_limit.png">

In [30]:

```
sns.set_theme(style="darkgrid", font='serif', context='talk')
class_0 = tr.loc[tr['default'] == 0]["limit"]
class_1 = tr.loc[tr['default'] == 1]["limit"]
plt.figure(figsize = (14,6))
plt.title('Credit Limit by Default', size=16)
sns.set_theme(style="darkgrid", font='serif', context='poster')
sns.distplot(class_1, kde=True, bins=200, color="blue", label="No")
sns.distplot(class_0, kde=True, bins=200, color="red", label="Yes")
plt.legend(title = 'default', loc='upper right', facecolor='white')
plt.show()
# plt.savefig("../images/credit_limit_by_default.png")
```



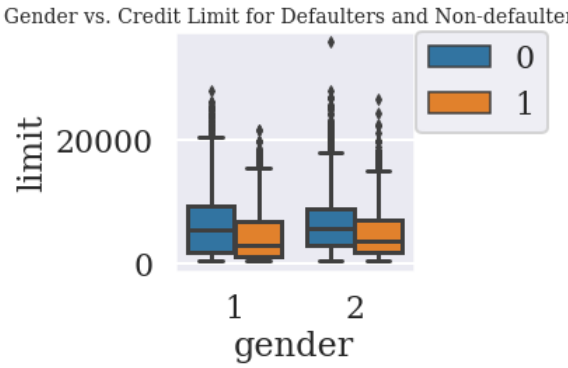
img src="../images/credit_limit_by_default.png">

In [31]:

```
sns.boxplot(x="gender", hue="default", y="limit", data=tr, palette="tab10")
plt.legend(bbox_to_anchor=(1.01, 1), borderaxespad=0)
plt.tight_layout()
plt.title("Gender vs. Credit Limit for Defaulters and Non-defaulters", size=14)
# plt.savefig("../images/boxplot4.png")
```

Out[31]:

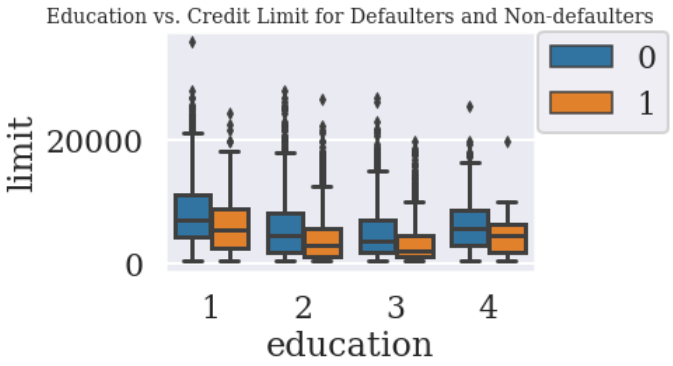
```
Text(0.5, 1.0, 'Gender vs. Credit Limit for Defaulters and Non-defaulters')
```



img src="../images/boxplot4.png">

In [32]:
```
sns.boxplot(x='education', hue='default', y='limit', data=tr, palette='tab10')
plt.legend(bbox_to_anchor=(1.01, 1), borderaxespad=0)
plt.tight_layout()
plt.title("Education vs. Credit Limit for Defaulters and Non-defaulters", size=14)
# plt.savefig("../images/boxplot5.png")
```

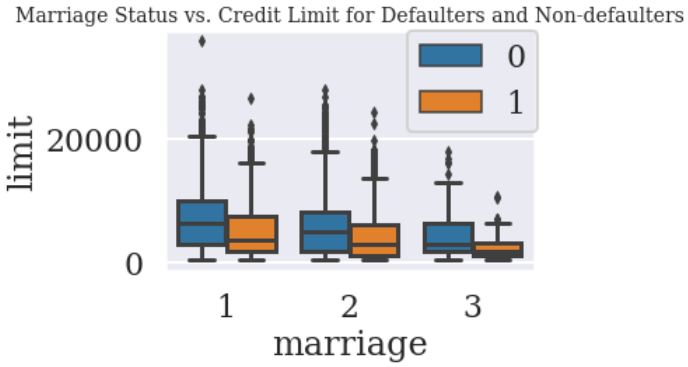
Out[32]:
```
Text(0.5, 1.0, 'Education vs. Credit Limit for Defaulters and Non-defaulters')
```



img src="../images/boxplot5.png">

In [33]:
```
sns.boxplot(x='marriage', hue='default', y='limit', data=tr, palette='tab10')
plt.legend(bbox_to_anchor=(1.01, 1), borderaxespad=0)
plt.tight_layout()
plt.title("Marriage Status vs. Credit Limit for Defaulters and Non-defaulters", size=14)
# plt.savefig("../images/boxplot6.png")
```

Out[33]:
```
Text(0.5, 1.0, 'Marriage Status vs. Credit Limit for Defaulters and Non-defaulters')
```



img src="../images/boxplot6.png">

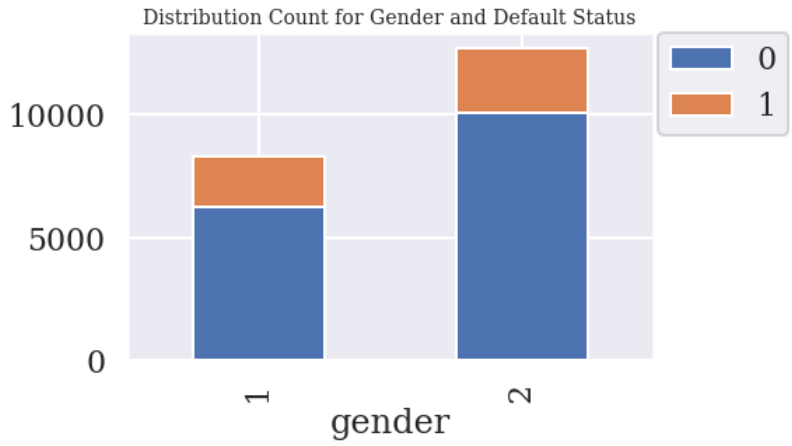
In [34]:
```
sex = tr.groupby(['gender', 'default']).size().unstack(1)
sex
```

Out[34]:

default	0	1
gender		
1	6272	2019
2	10072	2637

In [35]:
```
sns.set_theme(style="darkgrid", font='serif', context='poster')
sex.plot(kind="bar", stacked=True)
plt.legend(bbox_to_anchor=(1.01, 1), borderaxespad=0)
plt.title("Distribution Count for Gender and Default Status",size=14)
# plt.savefig("../images/stacked_bar.png")
```

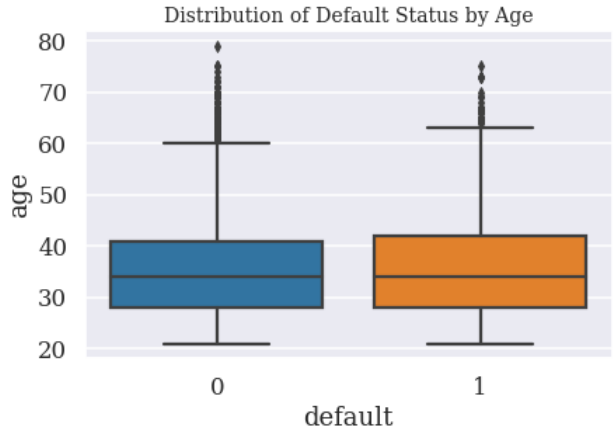
Out[35]:
```
Text(0.5, 1.0, 'Distribution Count for Gender and Default Status')
```



img src="../images/stacked_bar.png">

In [36]:
```
sns.set_theme(style="darkgrid", font='serif', context='talk')
sns.boxplot(x='default', y='age', data=tr, palette='tab10')
plt.title("Distribution of Default Status by Age", size=14)
# plt.savefig("../images/boxplot1.png")
```

Out[36]:
```
Text(0.5, 1.0, 'Distribution of Default Status by Age')
```



img src="../images/boxplot1.png">

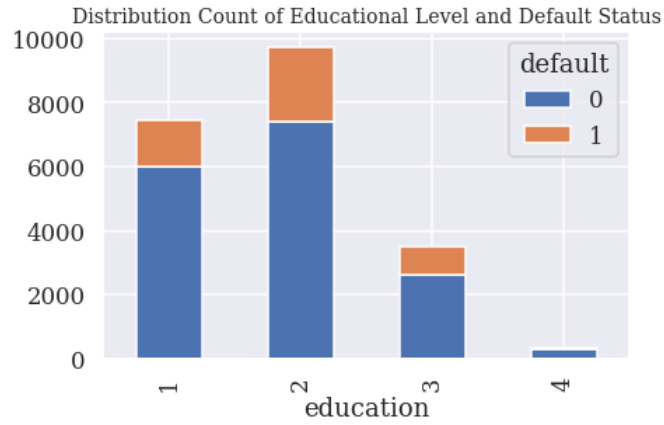
In [37]:
```
education = tr.groupby(['education', 'default']).size().unstack(1)
education
```

Out[37]:

default	0	1
education		
1	6013	1424
2	7408	2341
3	2626	866
4	297	25

In [38]:
```
education.plot(kind="bar", stacked=True)
plt.title("Distribution Count of Educational Level and Default Status", size=14)
# plt.savefig("../data/stacked_bar2.png")
```

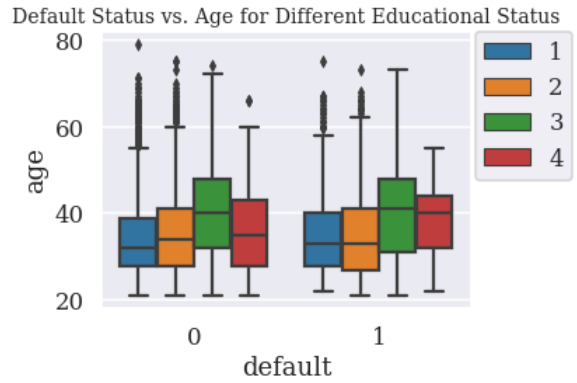
Out[38]:
```
Text(0.5, 1.0, 'Distribution Count of Educational Level and Default Status')
```



img src="../images/stacked_bar2.png">

In [39]:
```
sns.boxplot(x='default', y='age', hue='education', data=tr, palette='tab10')
plt.legend(bbox_to_anchor=(1.01, 1), borderaxespad=0)
plt.tight_layout()
plt.title("Default Status vs. Age for Different Educational Status", size=14)
# plt.savefig("../images/boxplot3.png")
```

Out[39]:
```
Text(0.5, 1.0, 'Default Status vs. Age for Different Educational Status')
```



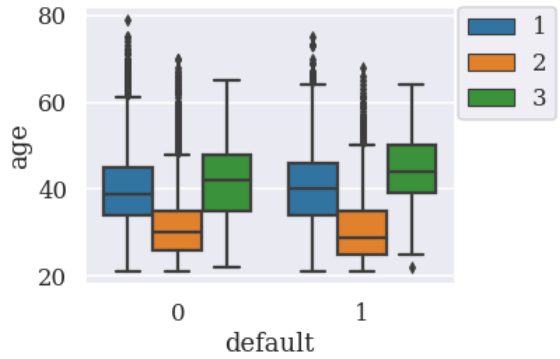
img src="../images/boxplot3.png">

In [40]:
```
marriage = tr.groupby(['marriage', 'default']).size().unstack(1)
marriage
```

Out[40]:

default	0	1
marriage		
1	7354	2258
2	8778	2336
3	212	62

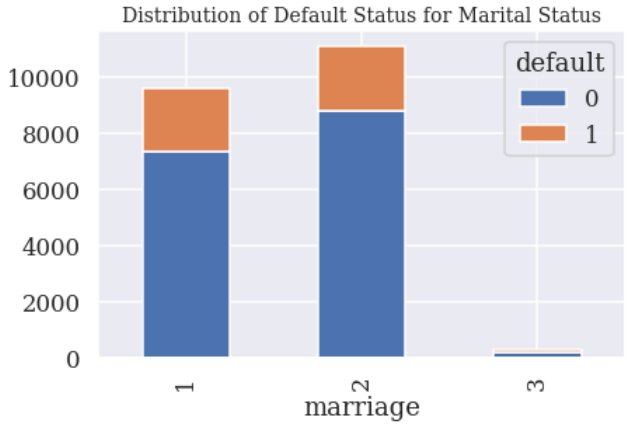
In [41]:
```
sns.boxplot(x='default', y='age', hue='marriage', data=tr, palette='tab10')
plt.legend(bbox_to_anchor=(1.01, 1), borderaxespad=0)
plt.tight_layout()
# plt.savefig("../images/boxplot2.png")
```



img src="../images/boxplot2.png">

In [42]:
```
marriage.plot(kind="bar", stacked=True)
plt.title("Distribution of Default Status for Marital Status", size=14)
# plt.savefig("../images/stacked_bar3.png")
```

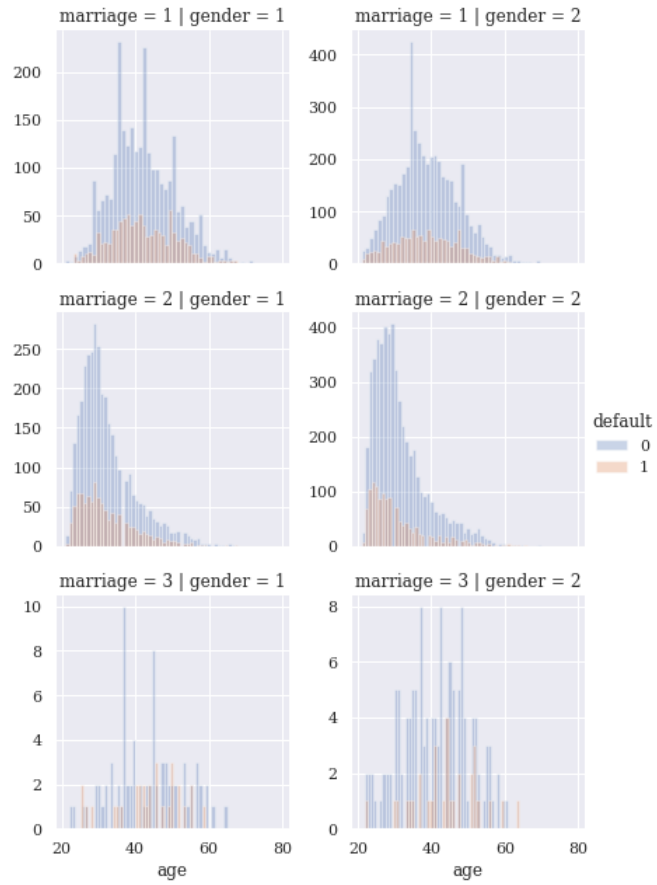
Out[42]: Text(0.5, 1.0, 'Distribution of Default Status for Marital Status')



img src="../images/stacked_bar3.png">

In [43]:
```
sns.set_theme(style="darkgrid", font='serif', context='notebook')
g = sns.FacetGrid(tr, col='gender', row="marriage", hue='default', sharey=False)
g.map(plt.hist, 'age', alpha=0.3, bins=50)
g.add_legend()
# g.savefig("../images/marriage_gender.png")
```

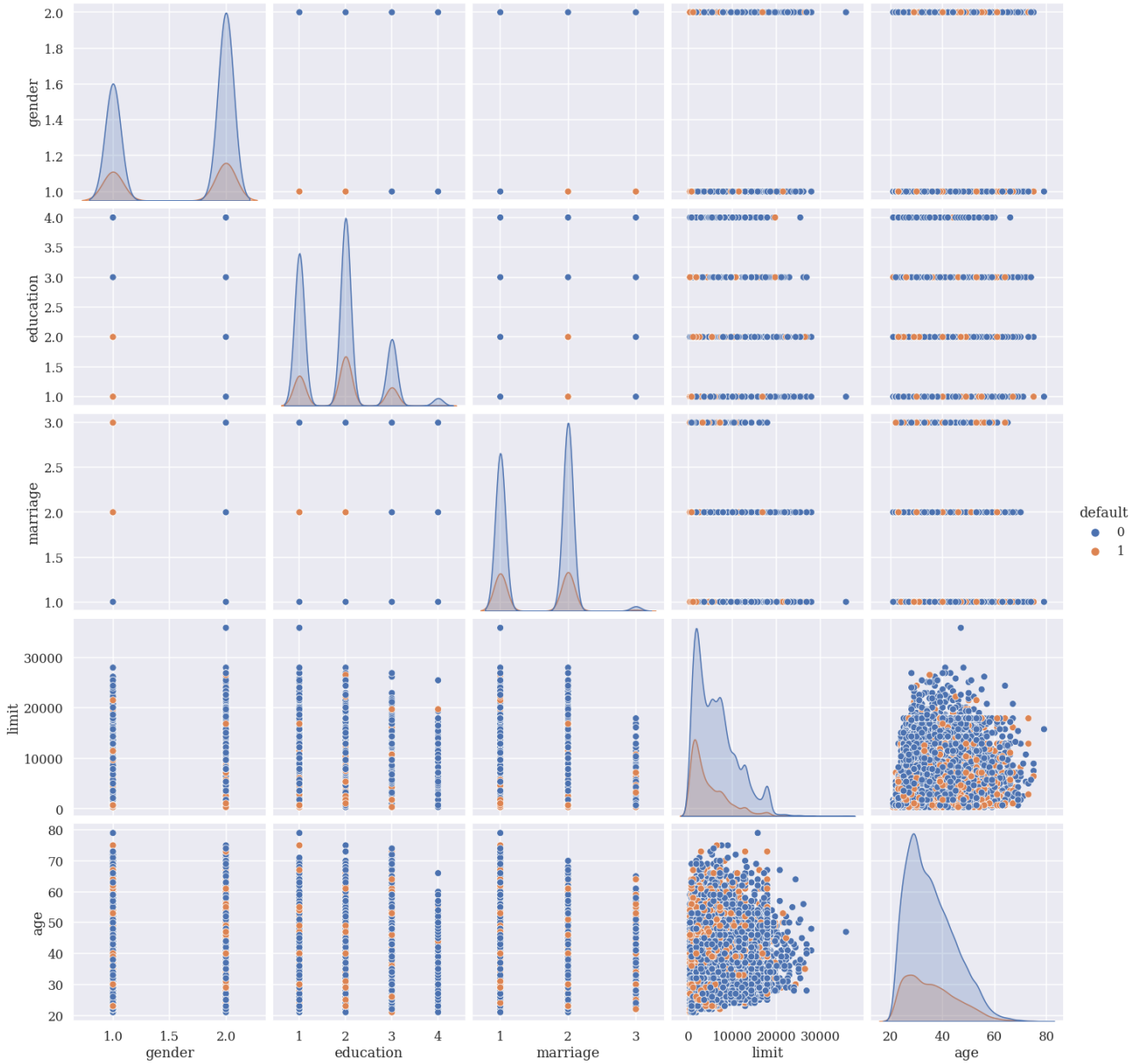
Out[43]: <seaborn.axisgrid.FacetGrid at 0x7fb0bb6c1fd0>



img src="../images/marriage_gender.png">

In [44]:

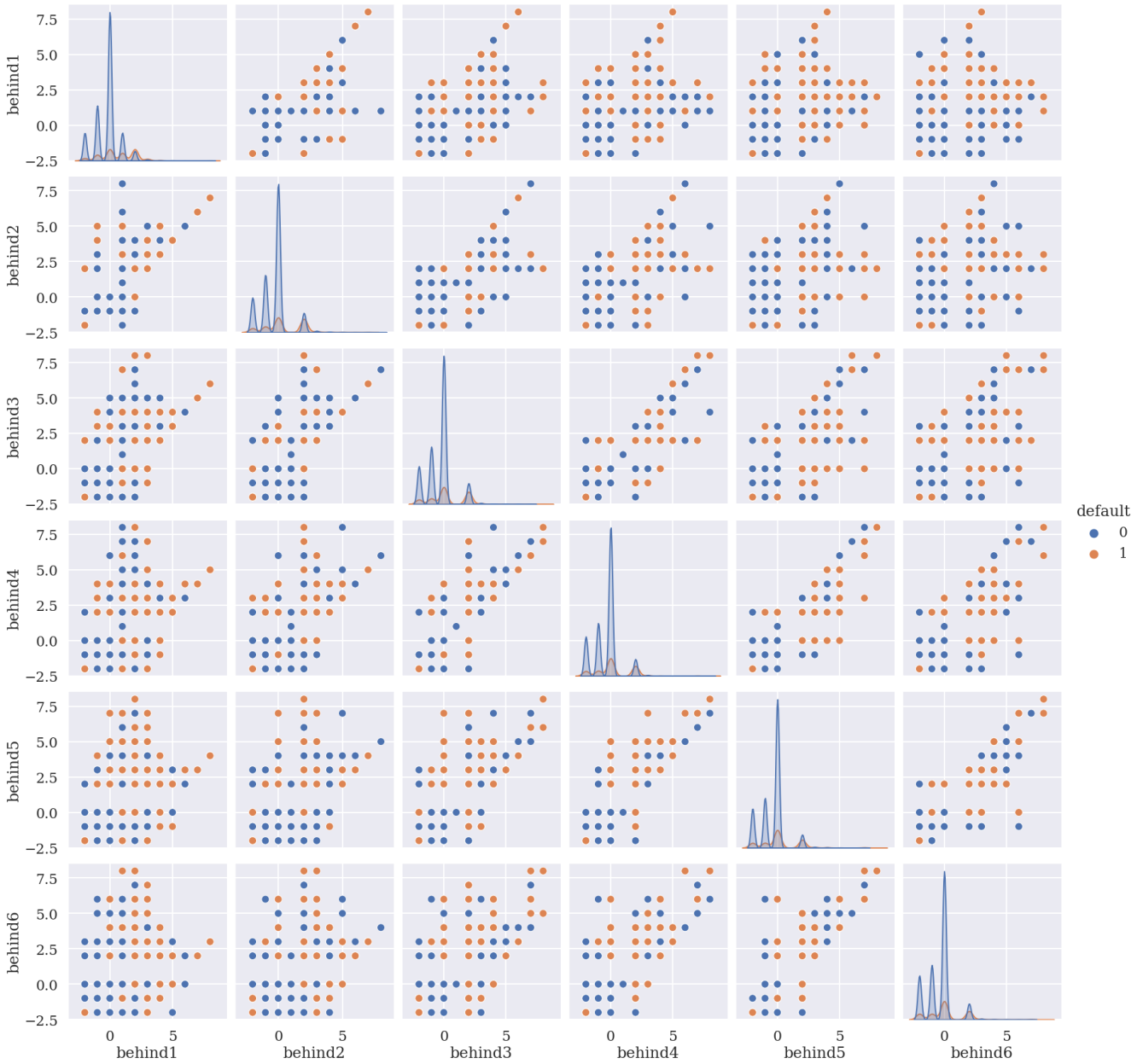
```
sns.set_theme(style="darkgrid", font='serif', context='talk')
g = sns.pairplot(tr[['gender', 'education', 'marriage', 'limit', 'age', 'default']],
                 hue="default",
                 diag_kind="kde",
                 size=4);
# g.savefig("../images/pairplot1.png")
```



img src="../images/pairplot1.png">

In [45]:

```
sns.set_theme(style="darkgrid", font='serif', context='poster')
g = sns.pairplot(tr[['behind1', 'behind2', 'behind3', 'behind4', 'behind5', 'behind6', 'default']],
                 hue="default",
                 diag_kind="kde",
                 size=4)
# g.savefig("../images/pairplot2.png")
```



img src="../images/pairplot2.png">

In [46]:

```
plt.style.use("fivethirtyeight")
sns.set_theme(style="darkgrid", font='serif', context='poster')
g = sns.pairplot(tr[['billed1', 'billed2', 'billed3', 'billed4', 'billed5', 'billed6', 'default']],
                 hue="default",
                 diag_kind="kde",
                 size=4)
# g.savefig("../images/pairplot3.png")
```

img src="../images/pairplot3.png">

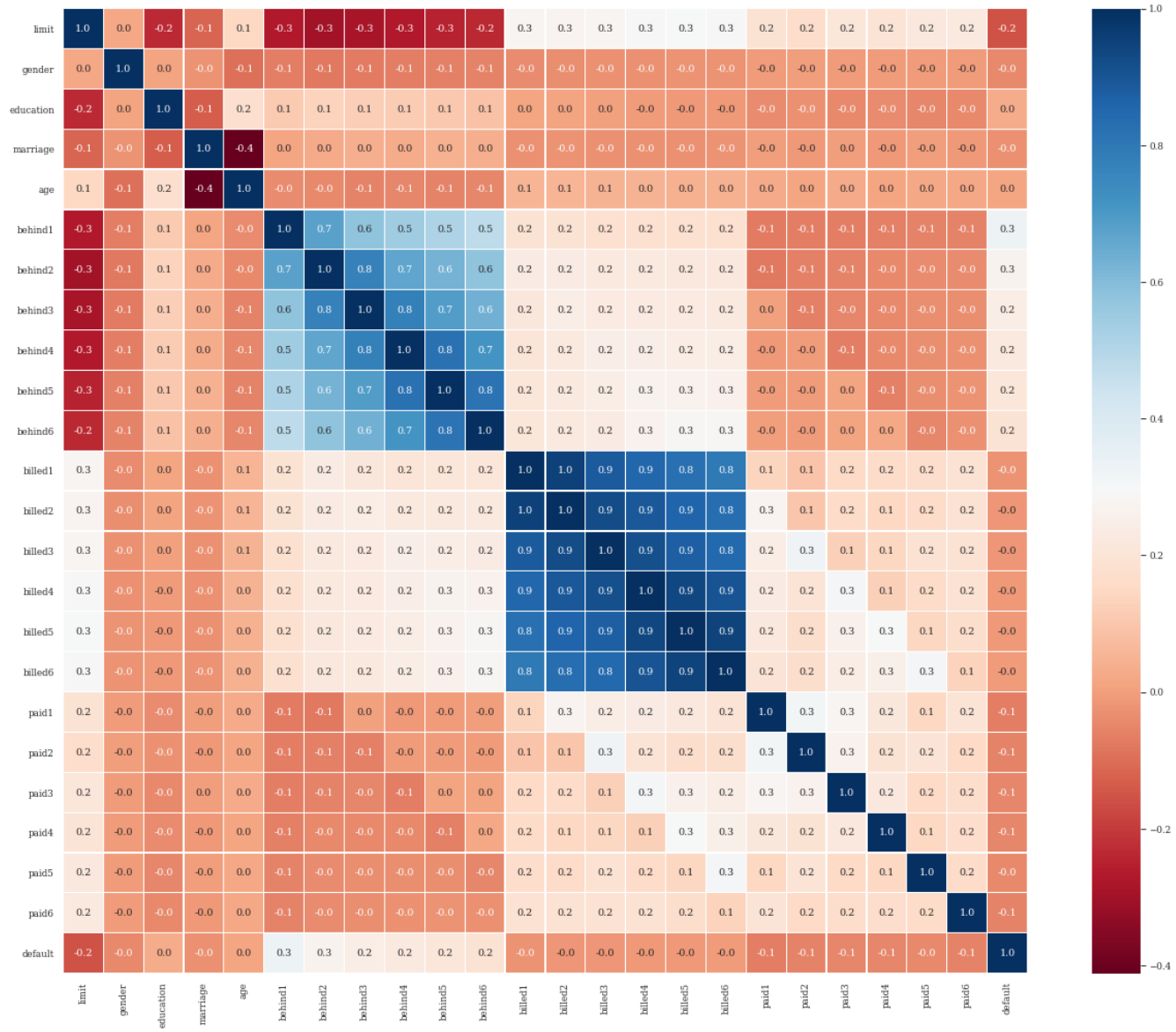
In [47]:

```
sns.set_theme(style="darkgrid", font='serif', context='talk')
g = sns.pairplot(tr[['paid1', 'paid2', 'paid3', 'paid4', 'paid5', 'paid6', 'default']],
                 hue="default",
                 diag_kind="kde",
                 size=4)
# g.savefig("../images/pairplot4.png")
```

img src="../images/pairplot4.png">

In [48]:

```
plt.style.use("fivethirtyeight")
sns.set_theme(style="darkgrid", font='serif', context='paper')
plt.figure(figsize = (20,16))
plt.title('Pearson Correlation of Features', y = 1.05, size = 20)
# mask = np.zeros(tr.corr().shape, dtype=bool)
# mask[np.triu_indices(len(mask))] = True
g = sns.heatmap(tr.corr(), cmap='RdBu', annot=True, square=True, linecolor='white', linewidths=0.2, fmt=".1f")
# plt.savefig("../images/correlation_matrix.png")
```



img src="../images/correlation_matrix.png">

In [45]:

```
sns.set_theme(style="darkgrid", font='serif', context='paper')
g = sns.JointGrid(data=tr, x="limit", y="age", hue='default')
g.plot(sns.scatterplot, sns.histplot)
```

Out[45]: <seaborn.axisgrid.JointGrid at 0x7f9c7d034af0>

Feature Engineering

In [60]:
```
tr.head()
```

Out[60]:

	limit	gender	education	marriage	age	behind1	behind2	behind3	behind4	behind5	behind6	billed1	billed2	billed3	billed4	billed
0	1789.97	2	2	1	44	0	0	0	0	0	0	1631.67	1500.21	1278.15	800.48	846.9
1	5727.92	2	3	1	46	-1	-1	-1	0	-1	-1	891.55	83.70	173.84	147.74	143.(
2	3579.95	2	2	1	47	-1	-1	-1	-1	-1	-2	238.64	238.64	0.00	224.46	-14.1
3	6085.91	2	2	1	29	0	0	0	0	0	0	2831.42	2240.15	2266.72	2287.69	1557.4
4	5369.92	2	1	2	33	-2	-2	-2	-2	-2	-2	873.26	961.11	1170.71	1197.81	995.2

In [49]:
```
data = [tr, val]

# create features for demographic variables
for d in data:
    d['age_bin'] = 0
    d.loc[((d['age'] > 20) & (d['age'] < 30)) , 'age_bin'] = 1
    d.loc[((d['age'] >= 30) & (d['age'] < 60)) , 'age_bin'] = 2
    d.loc[((d['age'] >= 60) & (d['age'] < 81)) , 'age_bin'] = 3

# feature for credit use percentage: fraction of estimated available balance based on what is billed per month
# (credit limit - monthly billed amount) / credit limit
for d in data:
    d['avail6'] = (d.limit - d.billed6) / d.limit
    d['avail5'] = (d.limit - d.billed5) / d.limit
    d['avail4'] = (d.limit - d.billed4) / d.limit
    d['avail3'] = (d.limit - d.billed3) / d.limit
    d['avail2'] = (d.limit - d.billed2) / d.limit
    d['avail1'] = (d.limit - d.billed1) / d.limit
    d['avg_av'] = (d.avail1 + d.avail2 + d.avail3 + d.avail4 + d.avail5 + d.avail6) / 6

# create a feature that indicates whether a client has had a delayed payment or not
def delayed_payment(d):
    if (d.behind1 > 0) or (d.behind2 > 0) or (d.behind3 > 0) or (d.behind4 > 0) or (d.behind5 > 0) or (d.behind6 > 0):
        return 1
    else:
        return 0
for d in data:
    d['delayed'] = d.apply(delayed_payment, axis=1)

# create feature for the total number of months with delayed payment status for a particular client
def total_months_with_delayed_payments(d):
```

```
    count = 0
    if (d.behind1 > 0):
        count += 1
    if (d.behind2 > 0):
        count += 1
    if (d.behind3 > 0):
        count += 1
    if (d.behind4 > 0):
        count += 1
    if (d.behind5 > 0):
        count += 1
    if (d.behind6 > 0):
        count += 1
    return count
for d in data:
    d['latemths'] = d.apply(total_months_with_delayed_payments, axis=1)

# the ratio of amount paid and amount billed
for d in data:
    d['pperb1'] = d.paid1 / d.billed2
    d['pperb2'] = d.paid2 / d.billed3
    d['pperb3'] = d.paid3 / d.billed4
    d['pperb4'] = d.paid4 / d.billed5
    d['pperb5'] = d.paid5 / d.billed6

# remove any infinity and NaN values
datasets = ['pperb1', 'pperb2', 'pperb3', 'pperb4', 'pperb5']
for data in datasets:
    tr.replace({data: {np.inf: 0, np.nan: 0}}, inplace=True)
    val.replace({data: {np.inf: 0, np.nan: 0}}, inplace=True)
```

In [50]:

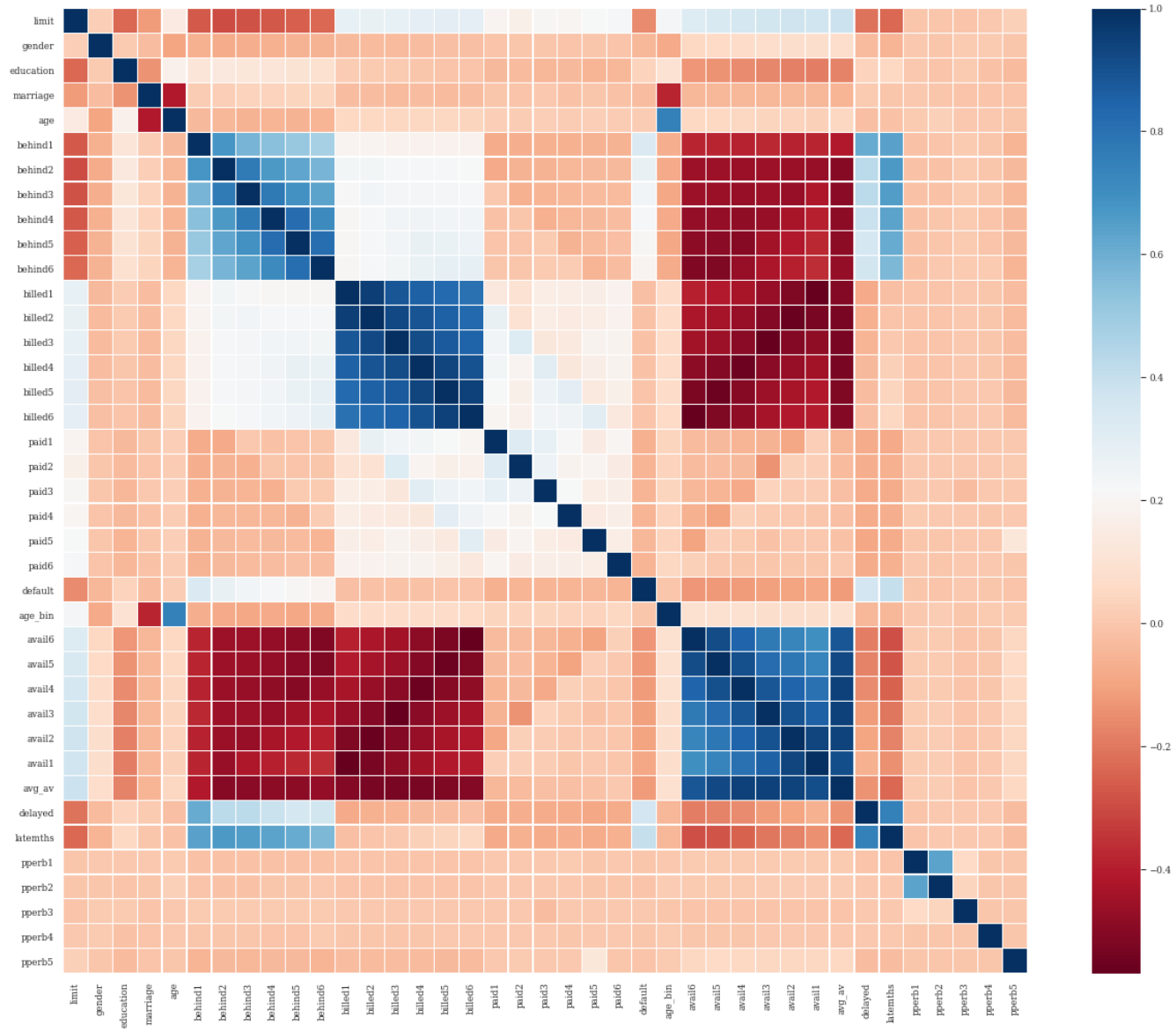
```
tr.head()
```

Out[50]:

	limit	gender	education	marriage	age	behind1	behind2	behind3	behind4	behind5	behind6	billed1	billed2	billed3	billed4	bille
0	1791.02	2	2	1	44	0	0	0	0	0	0	1632.63	1501.09	1278.90	800.95	847.
1	5731.27	2	3	1	46	-1	-1	-1	0	-1	-1	892.07	83.75	173.94	147.83	143
2	3582.05	2	2	1	47	-1	-1	-1	-1	-1	-2	238.78	238.78	0.00	224.59	-14
3	6089.48	2	2	1	29	0	0	0	0	0	0	2833.08	2241.47	2268.04	2289.04	1558
4	5373.07	2	1	2	33	-2	-2	-2	-2	-2	-2	873.77	961.67	1171.40	1198.52	995

In [51]:

```
plt.style.use("fivethirtyeight")
sns.set_theme(style="darkgrid", font='serif', context='paper')
plt.figure(figsize = (20,16))
plt.title('Pearson Correlation of Features', y = 1.05, size = 20)
g = sns.heatmap(tr.corr(), cmap='RdBu', square=True, linecolor='white', linewidths=0.2)
# plt.savefig("../images/correlation_matrix_2.png")
```



img src="../images/correlation_matrix_2.png">

In [65]:
```
tr.head()
```

Out[65]:

	limit	gender	education	marriage	age	behind1	behind2	behind3	behind4	behind5	behind6	billed1	billed2	billed3	billed4	billed
0	1789.97	2	2	1	44	0	0	0	0	0	0	1631.67	1500.21	1278.15	800.48	846.9
1	5727.92	2	3	1	46	-1	-1	-1	0	-1	-1	891.55	83.70	173.84	147.74	143.(
2	3579.95	2	2	1	47	-1	-1	-1	-1	-1	-2	238.64	238.64	0.00	224.46	-14.1
3	6085.91	2	2	1	29	0	0	0	0	0	0	2831.42	2240.15	2266.72	2287.69	1557.4
4	5369.92	2	1	2	33	-2	-2	-2	-2	-2	-2	873.26	961.11	1170.71	1197.81	995.2

In [69]:
```
pickle_out = open("../data/pickles/training_features.pickle","wb")
pickle.dump(tr, pickle_out)
pickle_out.close()
```

In [70]:
```
pickle_out = open("../data/pickles/validate_features.pickle","wb")
pickle.dump(val, pickle_out)
pickle_out.close()
```

Post-Feature Selection

In [82]:
```
tr.head()
```

Out[82]:

	limit	gender	education	marriage	age	behind1	behind2	behind3	behind4	behind5	behind6	billed1	billed2	billed3	billed4	billed
0	1790.26	2	2	1	44	0	0	0	0	0	0	1631.93	1500.45	1278.35	800.60	847.1
1	5728.83	2	3	1	46	-1	-1	-1	0	-1	-1	891.69	83.71	173.87	147.77	143.0
2	3580.52	2	2	1	47	-1	-1	-1	-1	-1	-2	238.68	238.68	0.00	224.50	-14.1
3	6086.88	2	2	1	29	0	0	0	0	0	0	2831.87	2240.51	2267.08	2288.06	1557.7
4	5370.78	2	1	2	33	-2	-2	-2	-2	-2	-2	873.40	961.26	1170.90	1198.01	995.3

In [83]:
```
train3 = tr[['limit', 'behind1', 'paid2', 'delayed', 'latemths', 'age', 'behind2', 'billed1', 'avg_av', 'avail1', 'defaul
validate3 = val[['limit', 'behind1', 'paid2', 'delayed', 'latemths', 'age', 'behind2', 'billed1', 'avg_av', 'avail1', 'de
```

In [84]:
```
train3.head()
```

Out[84]:

	limit	behind1	paid2	delayed	latemths	age	behind2	billed1	avg_av	avail1	default
0	1790.26	0	179.13	0	0	44	0	1631.93	0.344578	0.088440	0
1	5728.83	-1	173.87	0	0	46	-1	891.69	0.957227	0.844350	0
2	3580.52	-1	0.00	0	0	47	-1	238.68	0.968650	0.933339	1
3	6086.88	0	89.26	0	0	29	0	2831.87	0.650602	0.534758	0
4	5370.78	-2	1171.37	0	0	33	-2	873.40	0.836153	0.837379	0

In [85]:
```
validate3.head()
```

Out[85]:

	limit	behind1	paid2	delayed	latemths	age	behind2	billed1	avg_av	avail1	default
0	1074.16	0	71.61	0	0	25	0	317.38	0.602052	0.704532	0
1	5370.78	0	151.64	0	0	26	0	4895.86	0.293715	0.088427	0
2	2506.36	0	111.43	0	0	32	0	2510.73	0.005217	-0.001744	0
3	4654.68	0	64.74	0	0	49	0	740.38	0.883482	0.840939	0
4	1790.26	0	53.71	1	1	36	0	3373.85	0.188227	-0.884559	1

In [86]:
```
pickle_out = open("../data/training_model.pickle","wb")
pickle.dump(train3, pickle_out)
pickle_out.close()
```

In [87]:
```
pickle_out = open("../data/validate_model.pickle","wb")
pickle.dump(validate3, pickle_out)
pickle_out.close()
```

In [ ]:
```

```