código completo, com as indicações das imagens dos plots gerados:

```python
# **Carregamento da base de dados e declaração das bibliotecas usadas**

from google.colab import drive
drive.mount('/content/drive')

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import random
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report,
roc_auc_score

path = r'/content/drive/MyDrive/_credit_card_default_prediction_project/default of credit card
clients.xls' # Caminho do arquivo
defaults = pd.read_excel(path, engine = "xlrd") # Leitura da base de dados

# Renomeando as colunas
defaults.columns = [col for col in defaults.iloc[0, :]]
# Removendo a coluna ID e SEX
defaults.drop(columns = ["ID", "SEX"], axis = 1, inplace = True)
# Removendo a primeira linha que continha os nomes das colunas
defaults.drop(index = 0, axis = 0, inplace = True)
defaults.index = list(range(30000)) # Reindexando o dataframe

for x in defaults.columns: defaults[x] = defaults[x].apply(lambda x: int(x))

# Ajustando os valores da coluna EDUCATION
defaults["EDUCATION"] = defaults["EDUCATION"].apply(lambda x: 5 if x == 6 or x == 0 else x)
# Ajustando os valores da coluna MARRIAGE
defaults["MARRIAGE"] = defaults["MARRIAGE"].apply(lambda x: 3 if x == 0 else x)

# Função para calcular a proporção de adimplentes e inadimplentes
def proportion(database):
    index = sorted(list(set(database.iloc[:, 0])))
    columns = list(database.columns)
    adimplentes = database[database["default payment next month"] == 0]
    inadimplentes = database[database["default payment next month"] == 1]
    dataframe = pd.DataFrame(columns=["Adimplentes", "Inadimplentes"])

    for x in range(len(index)):
        total = len(database[database[columns[0]] == index[x]])
        adimplentes_count = len(adimplentes[adimplentes[columns[0]] == index[x]])
        inadimplentes_count = len(inadimplentes[inadimplentes[columns[0]] == index[x]])

        dataframe.loc[x] = {
            "Adimplentes": adimplentes_count / total,
            "Inadimplentes": inadimplentes_count / total
        }

    dataframe.index = index
```

```
    return dataframe

# **Estudo da base de dados**

## Educação

defaults_1 = defaults[~defaults["EDUCATION"].isin(range(1, 4))] # Filtrando os valores de
EDUCATION que não estão entre 1 e 3
defaults_2 = defaults[~defaults["EDUCATION"].isin([0, 4, 5, 6])] # Filtrando os valores de
EDUCATION que não estão entre 0, 4, 5 e 6

fig, (left, right) = plt.subplots(1, 2, figsize = (20, 8)) # Criando subplots e ajustando seu tamanho

# Gráfico 1 - Lado direito
sns.countplot(data = defaults_2, x = "EDUCATION", hue = "default payment next month", palette
= "viridis", ax = right)
right.set_title("Inadimplência e Adimplência por Educação", fontdict = {"fontsize": 15})
right.set_xlabel("Educação")
right.set_ylabel("Quantidade")
right.legend(title = "Pagamentos do próxmo mês", labels = ["Adimplente", "Inadimplente"])

right.spines["right"].set_visible(False)
right.spines["top"].set_visible(False)
right.set_xticks([0, 1, 2])
right.set_xticklabels(["Pós-Graduação", "Universidade", "Ensino Médio"])

# Gráfico 2 - Lado esquerdo
sns.countplot(data = defaults_1, x = "EDUCATION", hue = "default payment next month", palette
= "viridis", ax = left)
left.set_title("Inadimplência e Adimplência por Educação", fontdict = {"fontsize": 15})
left.set_xlabel("Educação")
left.set_ylabel("Quantidade")
left.legend(title = "Pagamentos do próxmo mês", labels = ["Adimplente", "Inadimplente"])

left.spines["right"].set_visible(False)
left.spines["top"].set_visible(False)
left.set_xticks([0, 1])
left.set_xticklabels(["Outros", "Não Conhecido"])

plt.show()  # Colar gráfico gerado aqui

# Gráfico 3 - Heatmap de proporção de adimplência e inadimplência por Educação
aux = proportion(defaults[["EDUCATION", "default payment next month"]])
aux.index = ["Escola", "Universidade", "Ensino médio", "Outros", "Desconhecido"]

plt.figure(figsize=(12, 9))
sns.heatmap(aux, vmin=0.0, vmax = 1.0, annot = True, cmap = "viridis", fmt = ".4f")
plt.title("Proporção de Adimplência e Inadimplência por Educação", fontdict = {"fontsize": 15})
plt.ylabel("Escolaridade")

plt.show()  # Colar gráfico gerado aqui

## Estado Social

plt.figure(figsize=(10, 6))
sns.countplot(data=defaults, x="MARRIAGE", hue="default payment next month",
palette="viridis")

plt.xticks(ticks=[0, 1, 2], labels=["Casado", "Solteiro", "Outros"])
plt.gca().spines["top"].set_visible(False)
```

```
plt.gca().spines["right"].set_visible(False)

plt.title("Inadimplência e Adimplência por Estado Civil", fontsize=15)
plt.xlabel("Estado Civil")
plt.ylabel("Quantidade")
plt.legend(title="Pagamento do Próximo mês", labels=["Adimplente", "Inadimplente"])

plt.show()  # Colar gráfico gerado aqui

# Gráfico 4 - Heatmap de proporção de adimplência e inadimplência por Estado Civil
aux = proportion(defaults[["MARRIAGE", "default payment next month"]])
aux.index = ["Casado", "Solteiro", "Outros"]

plt.figure(figsize=(12, 9))
sns.heatmap(aux, vmin=0.0, vmax = 1.0, annot = True, cmap = "viridis", fmt = ".4f")
plt.title("Proporção de Adimplêntes e Inadimplêntes por Estado Civil", fontdict = {"fontsize": 15})
plt.ylabel("Estado civil")

plt.show()  # Colar gráfico gerado aqui

## Idade

plt.figure(figsize = (17.5, 9))
sns.countplot(data = defaults, x = "AGE", hue = "default payment next month", palette = "viridis")

plt.gca().spines["top"].set_visible(False)
plt.gca().spines["right"].set_visible(False)

plt.title("Inadimplência e Adimplência por Idade")
plt.xlabel("Idade")
plt.ylabel("Quantidade")
plt.legend(title = "Pagamento do próximo mês", labels = ["Adimplente", "Inadimplente"])

plt.show()  # Colar gráfico gerado aqui

## Limite de Crédito

aux = defaults.copy()
aux['LIMIT_BAL_quantile'] = pd.qcut(defaults['LIMIT_BAL'], q=4, labels=["Até 50000", "De 50000
Até 140000", "De 140000 Até 240000", "Maior que 240000"])

plt.figure(figsize = (15, 8))
sns.countplot(data = aux, x = "LIMIT_BAL_quantile", hue = "default payment next month", palette
= "viridis")
plt.legend(title = "Pagamento do próximo mês", labels = ["Adimplente", "Inadimplente"])
plt.ylabel("Quantidade")
plt.xlabel("Limite de Crédito")
plt.title("Inadimplência e Adimplência por Limite de Crédito")

plt.gca().spines["top"].set_visible(False)
plt.gca().spines["right"].set_visible(False)

plt.show()  # Colar gráfico gerado aqui

## Status de Pagamento

fig, axes= plt.subplots(2, 3, figsize = (20, 12))
(upper_left, up, upper_right), (lower_left, down, lower_right) = axes
grafico = [upper_left, up, upper_right, lower_left, down, lower_right]
grafico.reverse()
```

```
coluna = list(range(0, 7))
coluna.remove(1)

meses = ["Setembro", "Agosto", "Julho", "Junho", "Maio", "Abril"]

for x in range(len(grafico)):
    sns.heatmap(data = proportion(defaults[[f"PAY_{coluna[x]}", "default payment next month"]]),
annot = True, cmap = "viridis", fmt = ".4f", ax = grafico[x])
    grafico[x].set_title("Status de pagamento de " + meses[x])

plt.show()  # Colar gráfico gerado aqui

## Valor

 pago

pagamento = [
    "BILL_AMT1",
    "BILL_AMT2",
    "BILL_AMT3",
    "BILL_AMT4",
    "BILL_AMT5",
    "BILL_AMT6"
]

index = 0
fig, ax = plt.subplots(2, 3, figsize = (20, 12))
for line in ax:
    for subplot in line:
        sns.histplot(data = defaults, x = pagamento[index], hue = "default payment next month",
palette = "viridis", ax = subplot)
        subplot.set_title(pagamento[index])
        index += 1

plt.show()  # Colar gráfico gerado aqui

## Conclusão

O estudo das variáveis educacionais, estado civil e idade demonstrou uma relação com a probabilidade de inadimplência.
O limite de crédito e o status de pagamento também são bons indicativos. Em gráficos futuros, vamos explorar o valor total da fatura e o montante pago.
```