# Multinomial Regression

**Instructions:**

Please share your answers filled in-line in the word document. Submit code separately wherever applicable.

Please ensure you update all the details:

**Name: Anandakrishnan k v ;;;;; Batch ID:** \_19042021

**Topic: Multinomial Regression.**

**Grading Guidelines:**

**1. An assignment submission is considered complete only when correct and executable code(s) are submitted along with the documentation explaining the method and results. Failing to submit either of those will be considered an invalid submission and will not be considered for evaluation.**

**2. Assignments submitted after the deadline will affect your grades.**

**Grading:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ans** | **Date** |  |  | **Ans** | **Date** |
| Correct | On time | A | 100 |  |  |
| 80% & above | On time | B | 85 | Correct | Late |
| 50% & above | On time | C | 75 | 80% & above | Late |
| 50% & below | On time | D | 65 | 50% & above | Late |
|  |  | E | 55 | 50% & below |  |
| Copied/No Submission |  | F | 45 |  |  |

* **Grade A: (>= 90):** When all assignments are submitted on or before the given deadline.
* **Grade B: (>= 80 and < 90):** 
  + When assignments are submitted on time but less than 80% of problems are completed.

(OR)

* + All assignments are submitted after the deadline.
* **Grade C: (>= 70 and < 80):** 
  + When assignments are submitted on time but less than 50% of the problems are completed.

(OR)

* + Less than 80% of problems in the assignments are submitted after the deadline.
* **Grade D: (>= 60 and < 70):**
  + Assignments submitted after the deadline and with 50% or less problems.
* **Grade E: (>= 50 and < 60):** 
  + Less than 30% of problems in the assignments are submitted after the deadline.

(OR)

* + Less than 30% of problems in the assignments are submitted before the deadline.
* **Grade F: (< 50):** No submission (or) malpractice.

**Hints:**

1. **Business Problem**
   1. **What is the business objective?**
   2. **Are there any constraints?**
2. **Work on each feature of the dataset to create a data dictionary as displayed in the below image:**



**2.1 Make a table as shown above and provide information about the features such as its data type and its relevance to the model building. And if not relevant, provide reasons and a description of the feature.**

**Using R and Python codes perform:**

1. **Data Pre-processing**

**3.1 Data Cleaning, Feature Engineering, etc.**

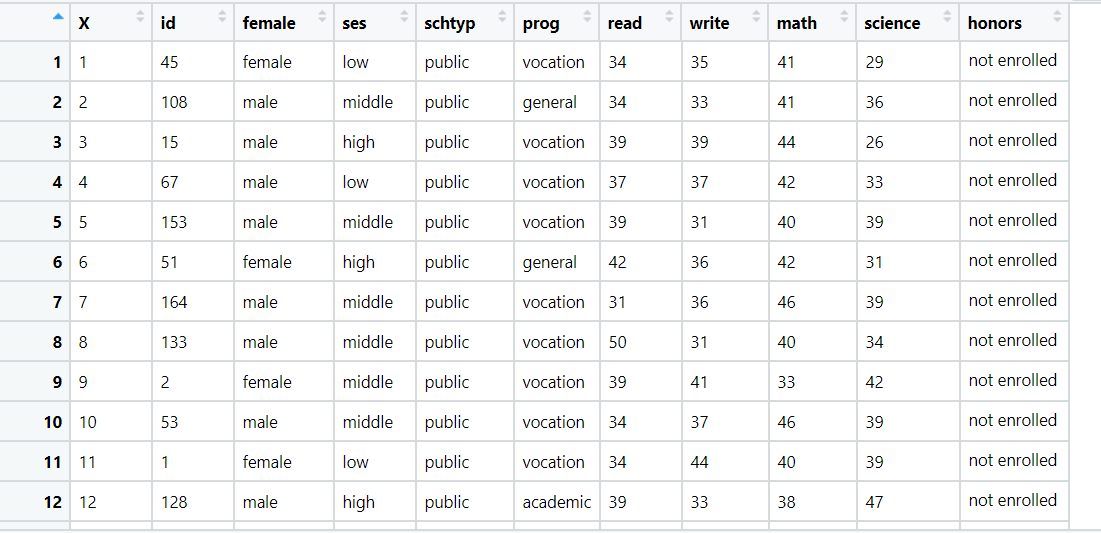
**3.2 Outlier treatment.**

1. **Exploratory Data Analysis (EDA):**
   1. **Summary.**
   2. **Univariate analysis.**
   3. **Bivariate analysis.**
2. **Model Building**
   1. **Build the model on the scaled data (try multiple options).**
   2. **Build a Multinomial Regression model.**
   3. **Train and test the model and compare accuracies by confusion matrix, ROC & AUC curves.**
   4. **Briefly explain the model output in the documentation.**
3. **Write about the benefits/impact of the solution - in what way does the business (client) benefit from the solution provided?**

**Problem Statement:**

A University would like to effectively classify their students based on the program they are enrolled in. Perform multinomial regression on the given dataset and provide insights (in the documentation).

1. **prog:** is a categorical variable indicating what type of program a student is in: “General” (1), “Academic” (2), or “Vocational” (3).
2. **Ses:** is a categorical variable indicating someone’s socioeconomic status: “Low” (1), “Middle” (2), and “High” (3).
3. **read, write, math, and science** are their scores on different tests.
4. **honors**: Whether they are an honor roll or not.



**PYTHON CODE:**

import pandas as pd

import seaborn as sns

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score

data = pd.read\_csv("C:\\Users\\karth\\Desktop\\DATA SCIENCE\\ASSIGNMENTS\\Datasets\_Multinomial\\mdata.csv")

data.head(10)

data.prog.value\_counts()

data.isna().sum()

from sklearn.preprocessing import LabelEncoder

data.describe()

lb = LabelEncoder()

data["female"] = lb.fit\_transform(data["female"])

data["ses"] = lb.fit\_transform(data["ses"])

data["schtyp"] = lb.fit\_transform(data["schtyp"])

data["honors"] = lb.fit\_transform(data["honors"])

data = data.iloc[:, [5 , 1, 2, 3, 4, 6, 7,8,9,10]]

# Boxplot of independent variable distribution for each category of prog

sns.set()

fig, axes = plt.subplots(3, 3, figsize=(15, 8))

fig.suptitle('loan')

axes[0,0].set\_title("id")

sns.boxplot(ax=axes[0,0],x = "prog", y = "id", data = data)

axes[0,1].set\_title("female")

sns.boxplot(ax=axes[0,1],x = "prog", y = "female", data = data)

axes[0,2].set\_title("ses")

sns.boxplot(ax=axes[0,2],x = "prog", y = "ses", data = data)

axes[1,0].set\_title("schtyp")

sns.boxplot(ax=axes[1,0],x = "prog", y = "schtyp", data = data)

axes[1,1].set\_title("read")

sns.boxplot(ax=axes[1,1],x = "prog", y = "read", data = data)

axes[1,2].set\_title("write")

sns.boxplot(ax=axes[1,2],x = "prog", y = "write", data = data)

axes[2,0].set\_title("math")

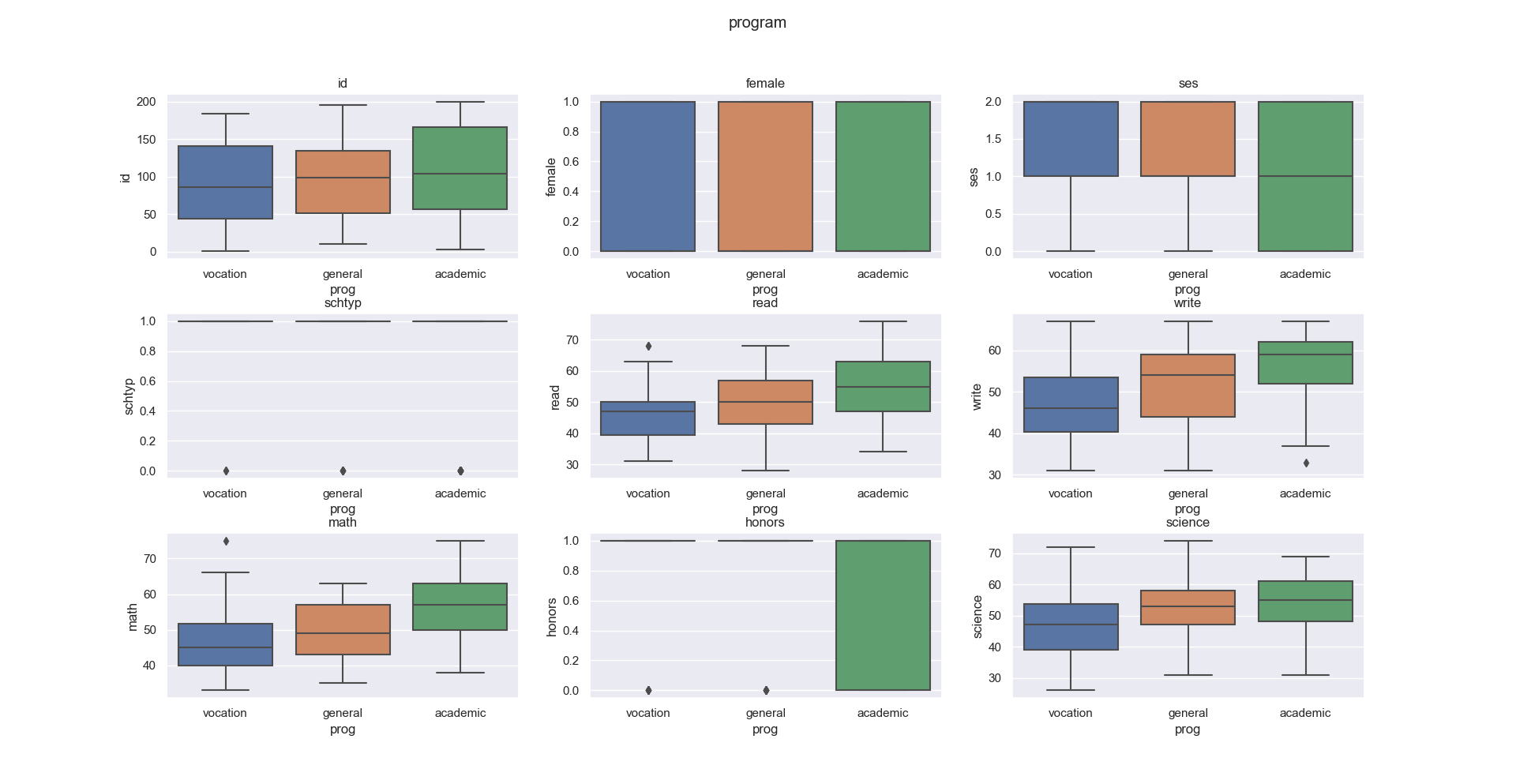
sns.boxplot(ax=axes[2,0],x = "prog", y = "math", data = data)

axes[2,1].set\_title("honors")

sns.boxplot(ax=axes[2,1],x = "prog", y = "honors", data = data)

axes[2,2].set\_title("science")

sns.boxplot(ax=axes[2,2],x = "prog", y = "science", data = data)



# Scatter plot for each categorical choice of prog

fig, axes = plt.subplots(3, 3, figsize=(15, 14))

fig.suptitle('loan')

axes[0,0].set\_title("id")

sns.stripplot(ax=axes[0,0],x = "prog", y = "id", jitter = True, data = data)

axes[0,1].set\_title("female")

sns.stripplot(ax=axes[0,1],x = "prog", y = "female", jitter = True, data = data)

axes[0,2].set\_title("ses")

sns.stripplot(ax=axes[0,2],x = "prog", y = "ses", jitter = True, data = data)

axes[1,0].set\_title("schtyp")

sns.stripplot(ax=axes[1,0],x = "prog", y = "schtyp", jitter = True, data = data)

axes[1,1].set\_title("read")

sns.stripplot(ax=axes[1,1],x = "prog", y = "read", jitter = True, data = data)

axes[1,2].set\_title("write")

sns.stripplot(ax=axes[1,2],x = "prog", y = "write", jitter = True, data = data)

axes[2,0].set\_title("math")

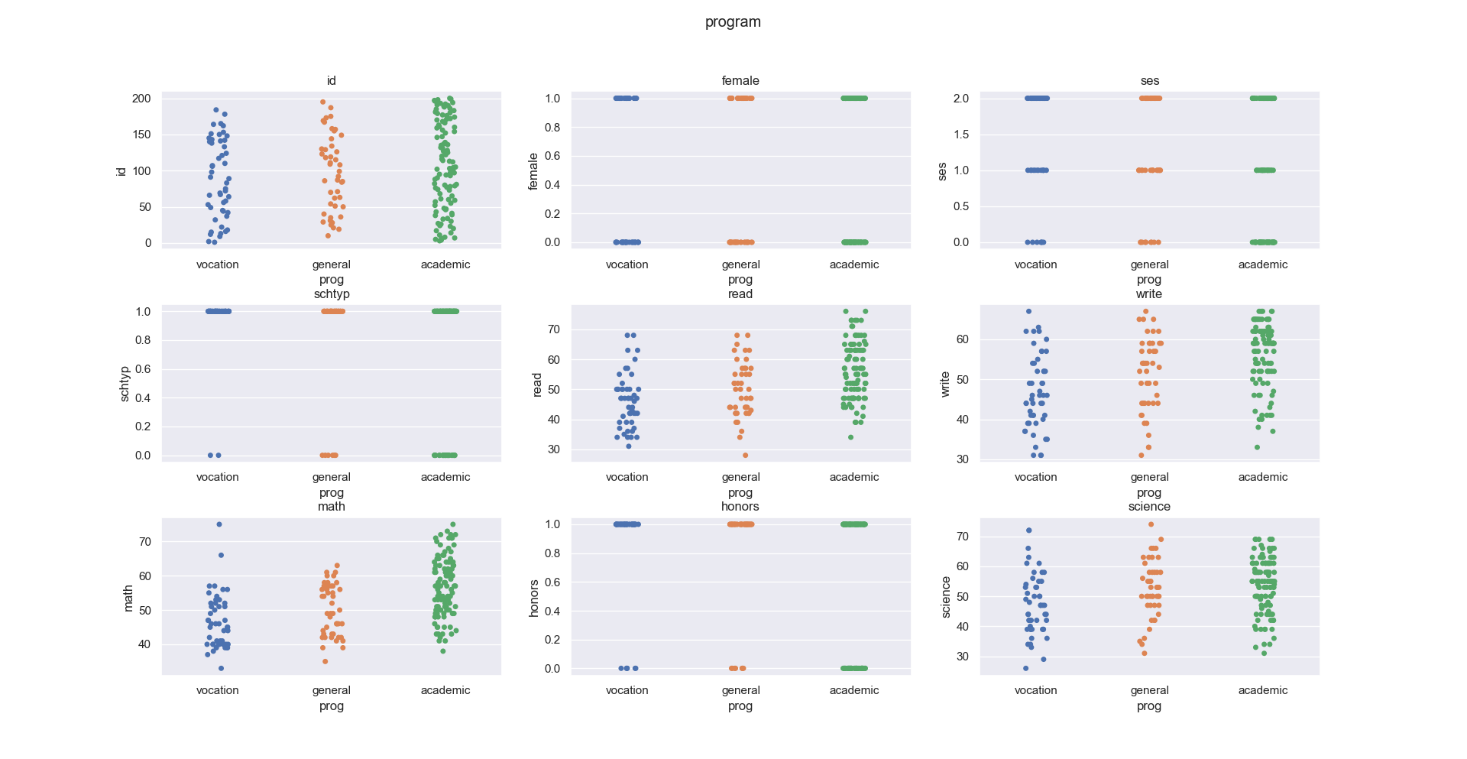
sns.stripplot(ax=axes[2,0],x = "prog", y = "math", jitter = True, data = data)

axes[2,1].set\_title("honors")

sns.stripplot(ax=axes[2,1],x = "prog", y = "honors", jitter = True, data = data)

axes[2,2].set\_title("science")

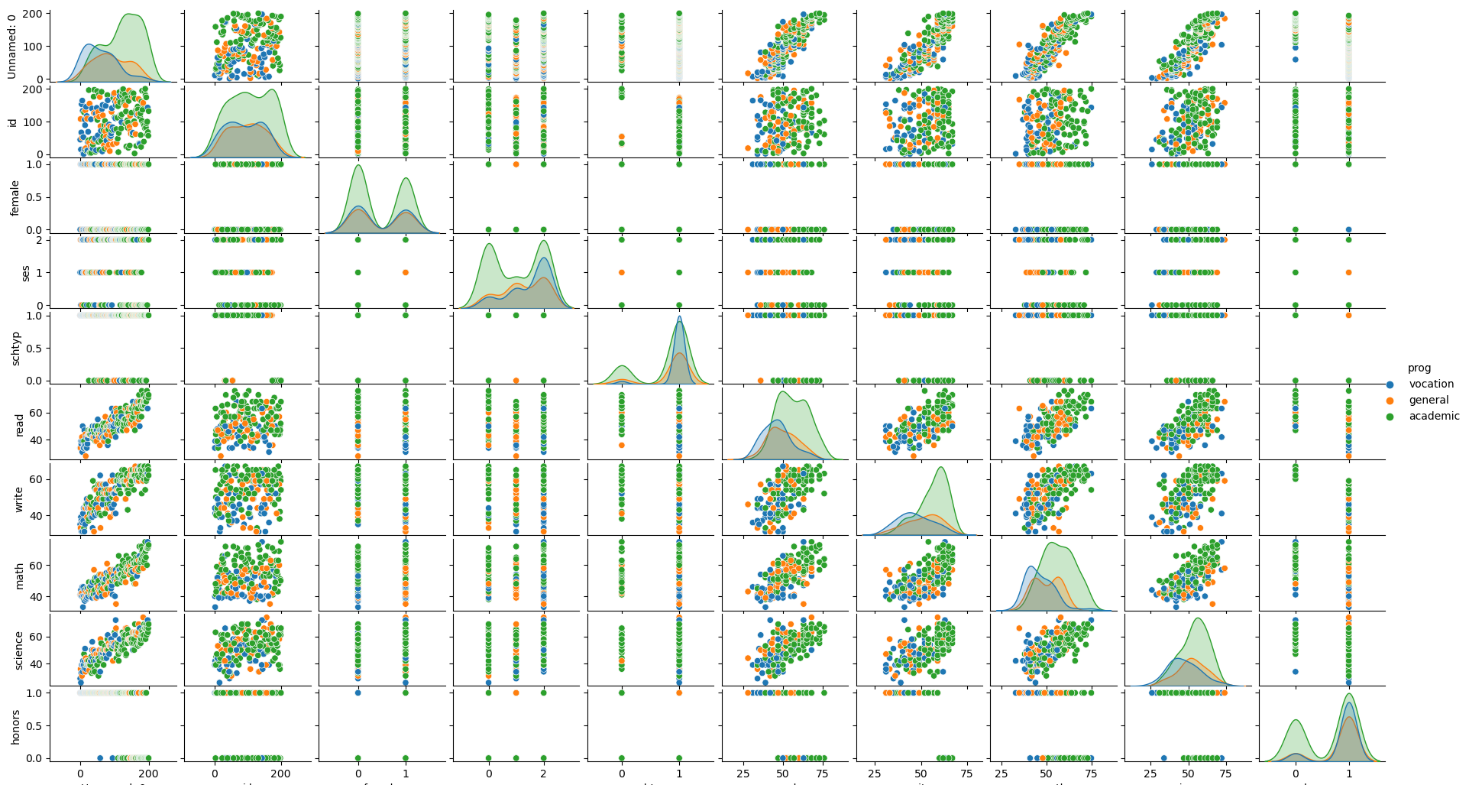
sns.stripplot(ax=axes[2,2],x = "prog", y = "science", jitter = True, data = data)



# Scatter plot between each possible pair of independent variable and also histogram for each independent variable

sns.pairplot(data) # Normal

sns.pairplot(data, hue = "prog") # With showing the category of each car choice in the scatter plot



# Correlation values between each independent features

data.corr()

train, test = train\_test\_split(data, test\_size = 0.2)

# ‘multinomial’ option is supported only by the ‘lbfgs’ and ‘newton-cg’ solvers

model = LogisticRegression(multi\_class = "multinomial", solver = "newton-cg").fit(train.iloc[:, 1:], train.iloc[:, 0])

help(LogisticRegression)

test\_predict = model.predict(test.iloc[:, 1:]) # Test predictions

# Test accuracy

accuracy\_score(test.iloc[:,0], test\_predict)

Out[436]: 0.60

train\_predict = model.predict(train.iloc[:, 1:]) # Train predictions

# Train accuracy

accuracy\_score(train.iloc[:,0], train\_predict)

Out[438]: 0.70

**Problem statement:**

You work for a consumer finance company which specializes in lending loans to urban customers. When the company receives a loan application, the company has to make a decision for loan approval based on the applicant’s profile. Two types of risks are associated with the bank’s decision:

• If the applicant is likely to repay the loan, then not approving the loan results in a loss of business to the company

• If the applicant is not likely to repay the loan, i.e. he/she is likely to default, then approving the loan may lead to a financial loss for the company

The data given below contains the information about past loan applicants and whether they ‘defaulted’4 or not. The aim is to identify patterns which indicate if a person is likely to default, which may be used for taking actions such as denying the loan, reducing the amount of loan, lending (to risky applicants) at a higher interest rate, etc.

In this case study, you will use EDA to understand how consumer attributes and loan attributes influence the tendency of default.

When a person applies for a loan, there are two types of decisions that could be taken by the company:

1. Loan accepted: If the company approves the loan, there are 3 possible scenarios described below:

o Fully paid: Applicant has fully paid the loan (the principal and the interest rate)

o Current: Applicant is in the process of paying the instalments, i.e. the tenure of the loan is not yet completed. These candidates are not labelled as 'defaulted'.

o Charged-off: Applicant has not paid the instalments in due time for a long period of time, i.e. he/she has defaulted on the loan

2. Loan rejected: The company had rejected the loan (because the candidate does not meet their requirements etc.). Since the loan was rejected, there is no transactional history of those applicants with the company and so this data is not available with the company (and thus in this dataset)

This company is the largest online loan marketplace, facilitating personal loans, business loans, and financing of medical procedures. Borrowers can easily access lower interest rate loans through an online interface.

Like most other lending companies, lending loans to ‘risky’ applicants is the largest source of financial loss (called credit loss). The credit loss is the amount of money lost by the lender when the borrower refuses to pay or runs away with the money owed. In other words, borrowers who default cause the largest amount of loss to the lenders. In this case, the customers labelled as 'charged-off' are the 'defaulters'.

If one is able to identify these risky loan applicants, then such loans can be reduced thereby cutting down the amount of credit loss.

In other words, the company wants to understand the driving factors (or driver variables) behind loan default, i.e. the variables which are strong indicators of default. The company can utilize this knowledge for its portfolio and risk assessment.

Perform Multinomial regression on the dataset in which loan\_status is the output (Y) variable and it has three levels in it.

A screenshot of a cell phone

Description automatically generated

**PYTHON CODE:**

import pandas as pd

import seaborn as sns

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score

from matplotlib import pyplot as plt

df = pd.read\_csv("C:\\Users\\karth\\Desktop\\DATA SCIENCE\\ASSIGNMENTS\\Datasets\_Multinomial\\loan.csv")

df.head(10)

data = df.iloc[:, [16,5 , 2, 7,8,9,12,13,14,20,24,25,31,32]]

data.isna().sum()

# discretizion

data['loan\_amnt'] = pd.cut(data['loan\_amnt'],3,labels=['low','average','high'])

data['annual\_inc'] = pd.cut(data['annual\_inc'],3,labels=['low','average','high'])

data['dti'] = pd.cut(data['dti'],3,labels=['low','average','high'])

data['installment'] = pd.cut(data['installment'],3,labels=['good','average','worst'])

data['revol\_bal'] = pd.cut(data['revol\_bal'],3,labels=['low','average','high'])

#labelencoder

from sklearn.preprocessing import LabelEncoder

w=data.describe()

lb = LabelEncoder()

data["term"] = lb.fit\_transform(data["term"])

data["loan\_amnt"] = lb.fit\_transform(data["loan\_amnt"])

data["grade"] = lb.fit\_transform(data["grade"])

data["sub\_grade"] = lb.fit\_transform(data["sub\_grade"])

data["home\_ownership"] = lb.fit\_transform(data["home\_ownership"])

data["purpose"] = lb.fit\_transform(data["purpose"])

data["verification\_status"] = lb.fit\_transform(data["verification\_status"])

data["annual\_inc"] = lb.fit\_transform(data["annual\_inc"])

data["dti"] = lb.fit\_transform(data["dti"])

data["installment"] = lb.fit\_transform(data["installment"])

data["revol\_bal"] = lb.fit\_transform(data["revol\_bal"])

# Boxplot of independent variable distribution for each category of prog

sns.set()

fig, axes = plt.subplots(3, 5, figsize=(18, 18))

fig.suptitle('loan')

axes[0,0].set\_title("term")

sns.boxplot(ax=axes[0,0],x = "loan\_status", y = "term", data = data)

axes[0,1].set\_title("dti")

sns.boxplot(ax=axes[0,1],x = "loan\_status", y = "dti", data = data)

axes[0,2].set\_title("installment")

sns.boxplot(ax=axes[0,2],x = "loan\_status", y = "installment", data = data)

axes[0,3].set\_title("grade")

sns.boxplot(ax=axes[0,3],x = "loan\_status", y = "grade", data = data)

axes[0,4].set\_title("sub\_grade")

sns.boxplot(ax=axes[0,4],x = "loan\_status", y = "sub\_grade", data = data)

axes[1,0].set\_title("home\_ownership")

sns.boxplot(ax=axes[1,0],x = "loan\_status", y = "home\_ownership", data = data)

axes[1,1].set\_title("annual\_inc")

sns.boxplot(ax=axes[1,1],x = "loan\_status", y = "annual\_inc", data = data)

axes[1,2].set\_title("verification\_status")

sns.boxplot(ax=axes[1,2],x = "loan\_status", y = "verification\_status", data = data)

axes[1,3].set\_title("revol\_bal")

sns.boxplot(ax=axes[1,3],x = "loan\_status", y = "revol\_bal", data = data)

axes[1,4].set\_title("purpose")

sns.boxplot(ax=axes[1,4],x = "loan\_status", y = "purpose", data = data)

axes[2,0].set\_title("delinq\_2yrs")

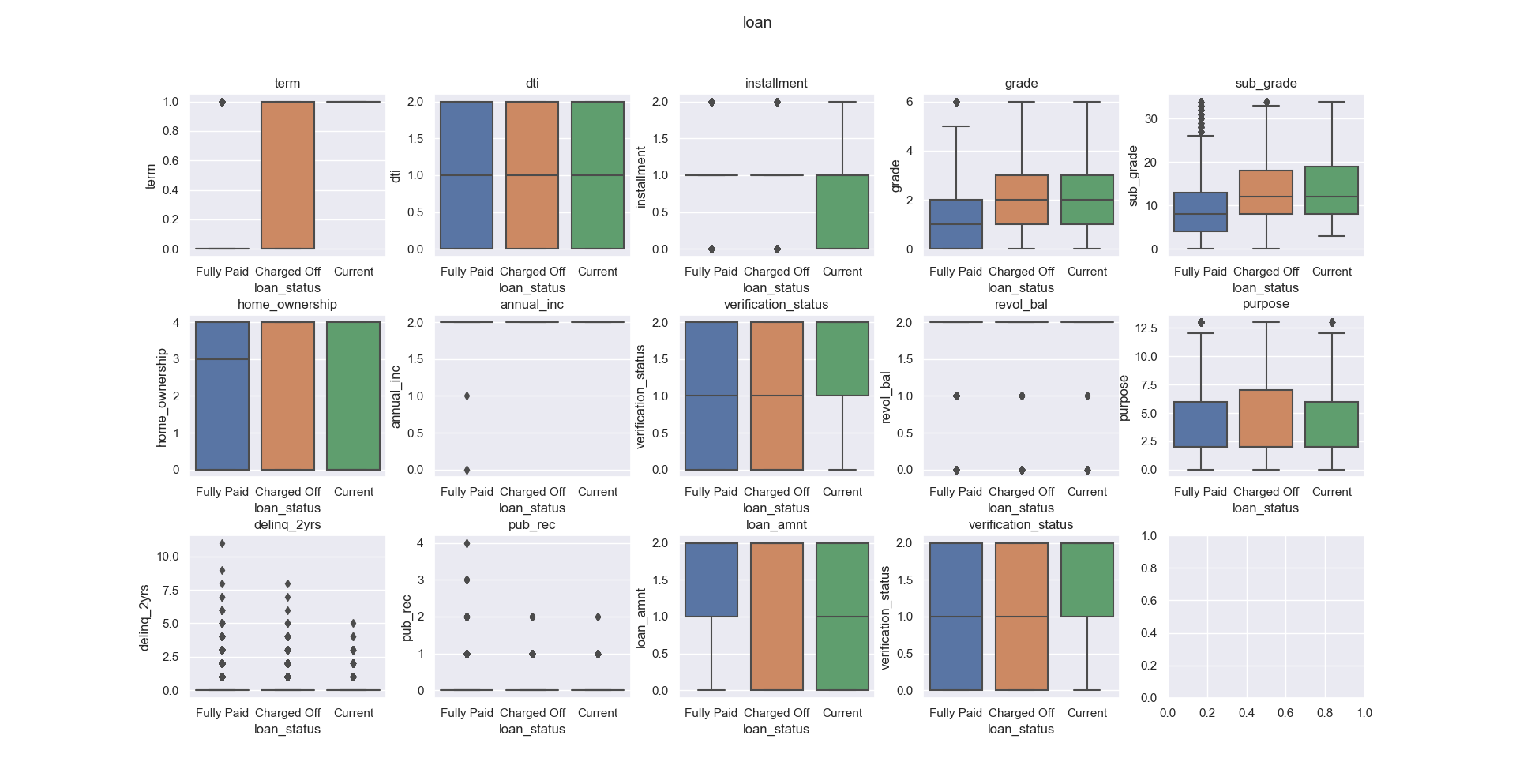
sns.boxplot(ax=axes[2,0],x = "loan\_status", y = "delinq\_2yrs", data = data)

axes[2,1].set\_title("pub\_rec")

sns.boxplot(ax=axes[2,1],x = "loan\_status", y = "pub\_rec", data = data)

axes[2,2].set\_title("loan\_amnt")

sns.boxplot(ax=axes[2,2],x = "loan\_status", y = "loan\_amnt", data = data)



# Scatter plot for each categorical choice of car

fig, axes = plt.subplots(3, 5, figsize=(18, 18))

fig.suptitle('loan')

axes[0,0].set\_title("term")

sns.stripplot(ax=axes[0,0],x = "loan\_status", y = "term", jitter = True, data = data)

axes[0,1].set\_title("dti")

sns.stripplot(ax=axes[0,1],x = "loan\_status", y = "dti", jitter = True, data = data)

axes[0,2].set\_title("installment")

sns.stripplot(ax=axes[0,2],x = "loan\_status", y = "installment", jitter = True, data = data)

axes[0,3].set\_title("grade")

sns.stripplot(ax=axes[0,3],x = "loan\_status", y = "grade", jitter = True, data = data)

axes[0,4].set\_title("sub\_grade")

sns.stripplot(ax=axes[0,4],x = "loan\_status", y = "sub\_grade", jitter = True, data = data)

axes[1,0].set\_title("home\_ownership")

sns.stripplot(ax=axes[1,0],x = "loan\_status", y = "home\_ownership", jitter = True, data = data)

axes[1,1].set\_title("annual\_inc")

sns.stripplot(ax=axes[1,1],x = "loan\_status", y = "annual\_inc", jitter = True, data = data)

axes[1,2].set\_title("verification\_status")

sns.stripplot(ax=axes[1,2],x = "loan\_status", y = "verification\_status", jitter = True, data = data)

axes[1,3].set\_title("revol\_bal")

sns.stripplot(ax=axes[1,3],x = "loan\_status", y = "revol\_bal", jitter = True, data = data)

axes[1,4].set\_title("purpose")

sns.stripplot(ax=axes[1,4],x = "loan\_status", y = "purpose", jitter = True, data = data)

axes[2,0].set\_title("delinq\_2yrs")

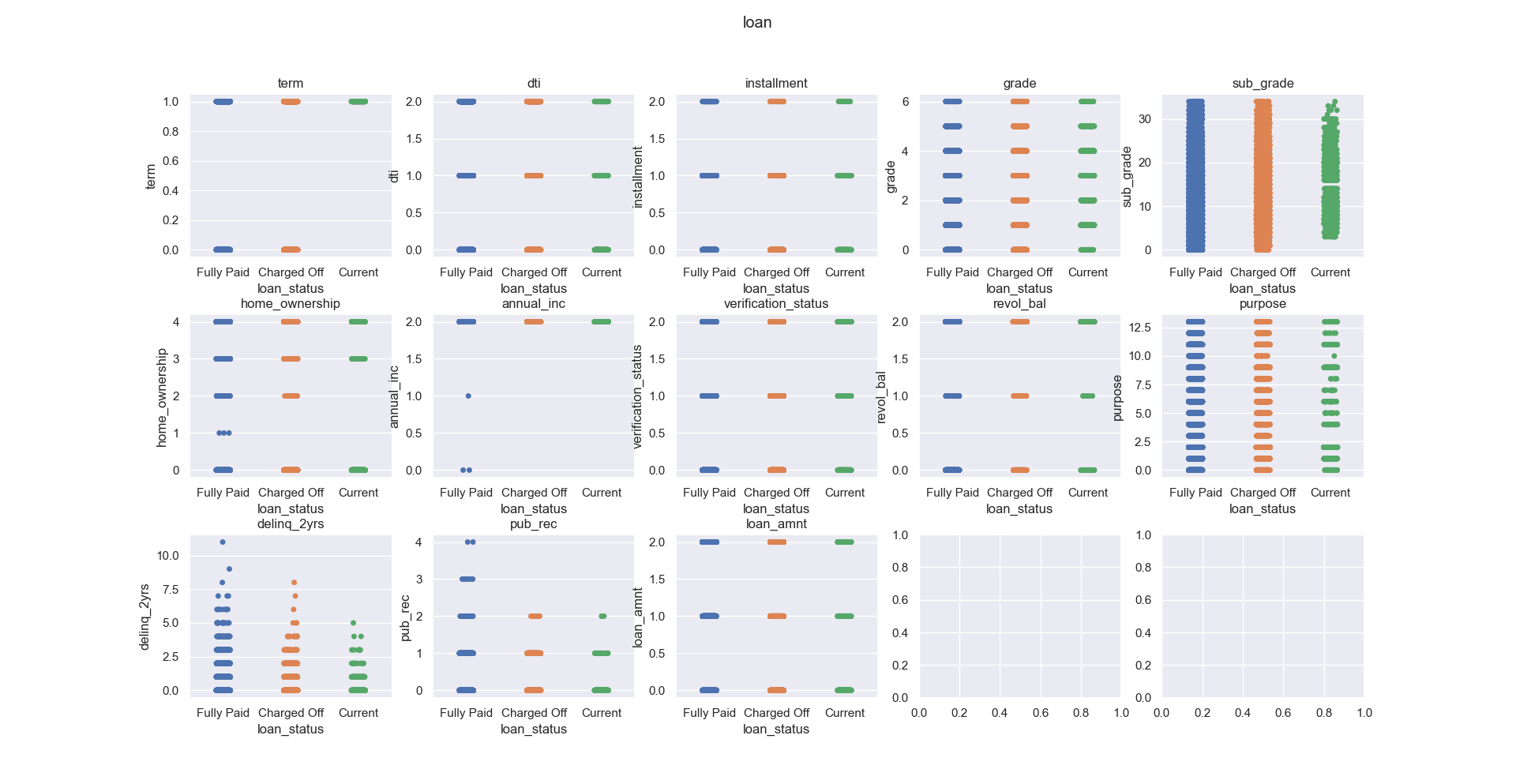
sns.stripplot(ax=axes[2,0],x = "loan\_status", y = "delinq\_2yrs", jitter = True, data = data)

axes[2,1].set\_title("pub\_rec")

sns.stripplot(ax=axes[2,1],x = "loan\_status", y = "pub\_rec", jitter = True, data = data)

axes[2,2].set\_title("loan\_amnt")

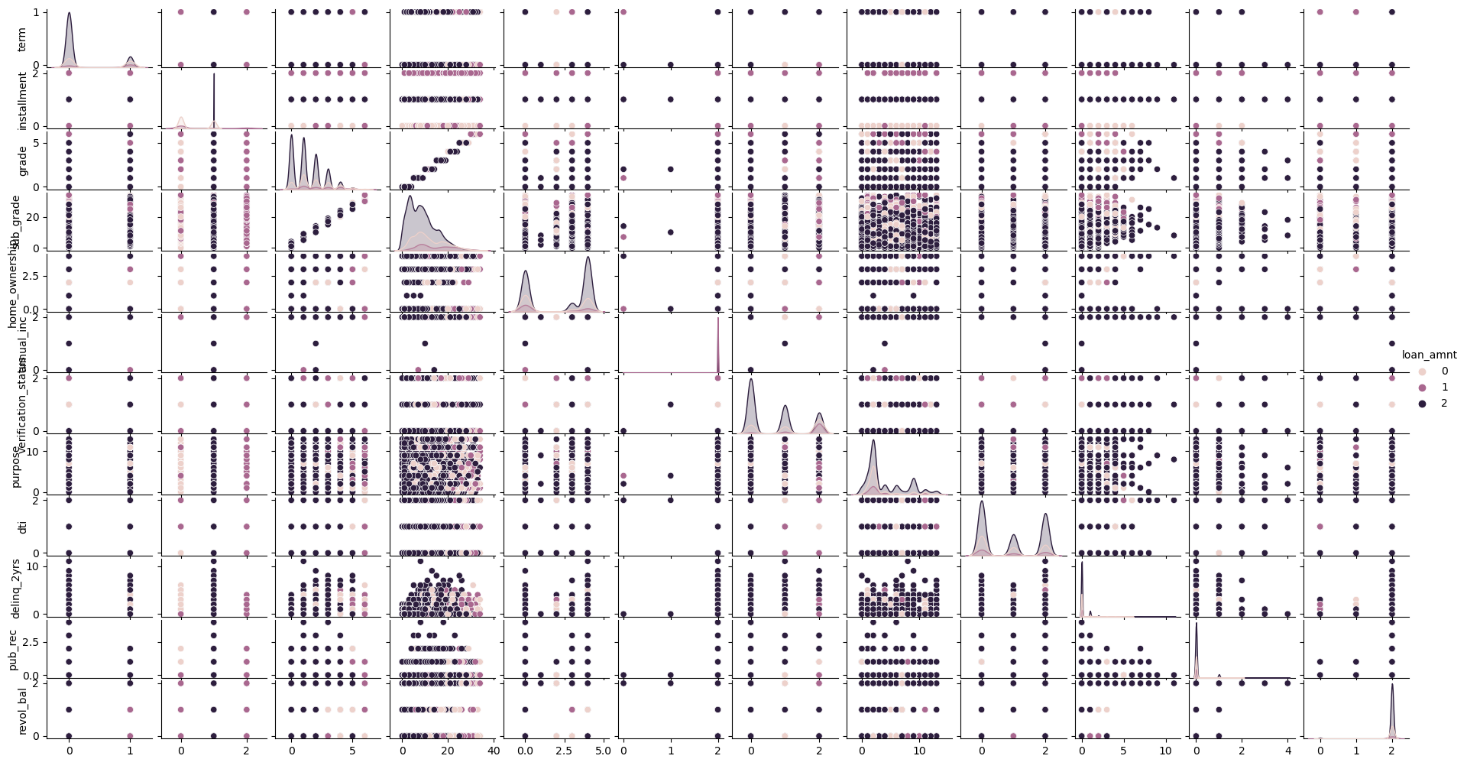
sns.stripplot(ax=axes[2,2],x = "loan\_status", y = "loan\_amnt", jitter = True, data = data)



# Scatter plot between each possible pair of independent variable and also histogram for each independent variable

sns.pairplot(data) # Normal

sns.pairplot(data, hue = "loan\_amnt") # With showing the category of each car choice in the scatter plot



# Correlation values between each independent features

s=data.corr()

train, test = train\_test\_split(data, test\_size = 0.3)

# ‘multinomial’ option is supported only by the ‘lbfgs’ and ‘newton-cg’ solvers

model = LogisticRegression(multi\_class = "multinomial", solver = "newton-cg").fit(train.iloc[:, 1:], train.iloc[:, 0])

help(LogisticRegression)

test\_predict = model.predict(test.iloc[:, 1:]) # Test predictions

# Test accuracy

accuracy\_score(test.iloc[:,0], test\_predict)

Out[89]: 0.8274588788183954

train\_predict = model.predict(train.iloc[:, 1:]) # Train predictions

# Train accuracy

accuracy\_score(train.iloc[:,0], train\_predict)

Out[91]: 0.8300061148879536