SC1015

Mini Project

B137 Team 4

Puah Rong Qi (U2221477H)

Kiersten Yeo Shu Xian (U2220969G)

Manasarow Gunasekaran (U2222761C)

MOTIVATION

01

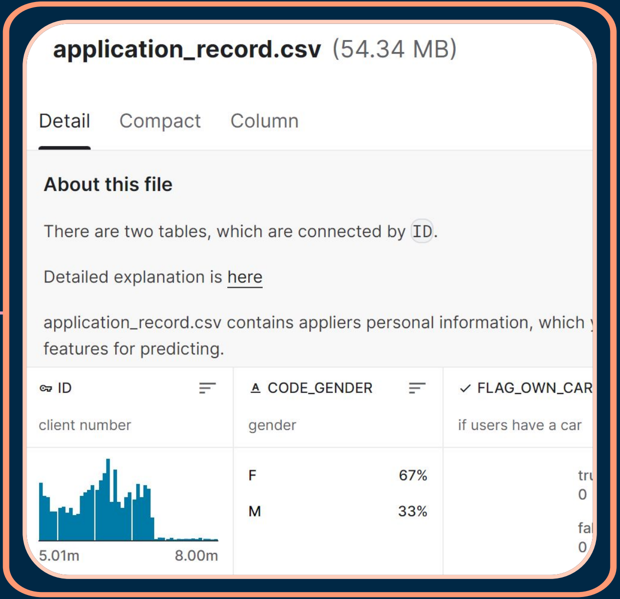
PROBLEM

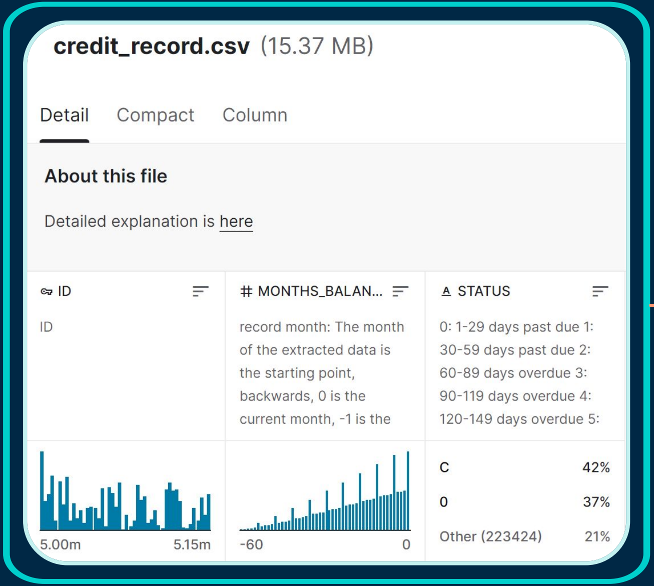
- When issuing credit cards, the personal information of applicants is used to decide whether they should be issued a credit card

- One important factor is the applicant's loan payment history

Which variable affects a person's ability to pay off loans on time the most?

DATASET

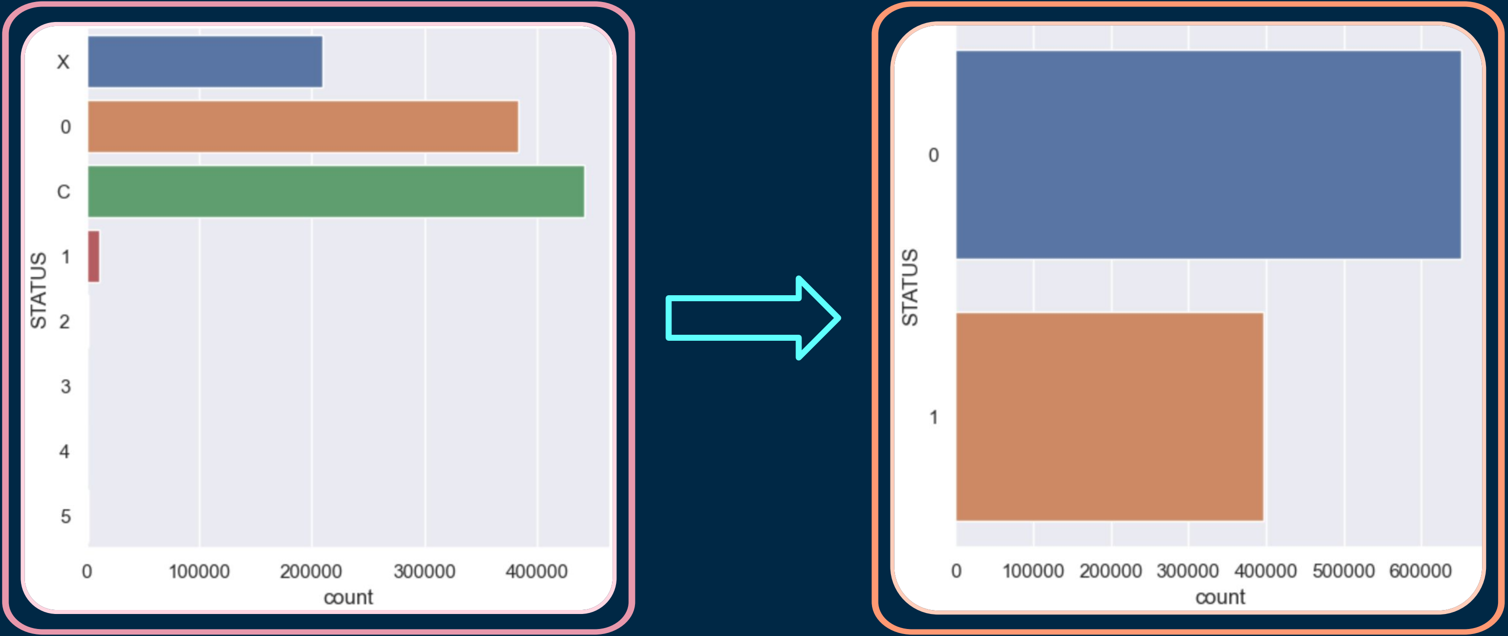

application_record.csv (54.34 MB)
Detail
Compact
Column
About this file
There are two tables, which are connected by ID.
Detailed explanation is here
application_record.csv contains appliers personal information, which
features for predicting.
ID
client number
5.01m
8.00m
CODE_GENDER
gender
F 67%
M 33%
FLAG_OWN_CAR
if users have a car


credit_record.csv (15.37 MB)
Detail
Compact
Column
About this file
Detailed explanation is here
ID
ID
5.00m
5.15m
MONTHS_BALAN...
record month: The month of the extracted data is the starting point, backwards, 0 is the current month, -1 is the
-60
0
STATUS
0: 1-29 days past due 1: 30-59 days past due 2: 60-89 days overdue 3: 90-119 days overdue 4: 120-149 days overdue 5:
C 42%
0 37%
Other (223424) 21%

DATA PREPARATION

02

DATA PREPARATION



Changed all the values of
the variable 'STATUS' to numbers

DATA PREPARATION

NULL values

Filled up NULL values
in ‘OCCUPATION_TYPE’

```python
# Filling the NULL values in Occupation Type in ar
ar['OCCUPATION_TYPE'].fillna(value="NA", inplace= True)
```

‘ID’

Combined rows
with the same IDs

```python
# Combining all the repeated IDs and averaging out STATUS
cr = cr.groupby(['ID'])['STATUS'].agg('mean')
```

Two CSV files

Merged the two dataframes,
removed IDs not in both files

```python
# Merging the 2 dataframes and removing all different IDs
merged_df = pd.merge(cr, ar, on="ID", how="inner")
# Rounding up STATUS to whole nummber
merged_df["STATUS"] = np.ceil(merged_df["STATUS"])
```

EXPLORATORY DATA ANALYSIS

03

VARIABLE SELECTION 1

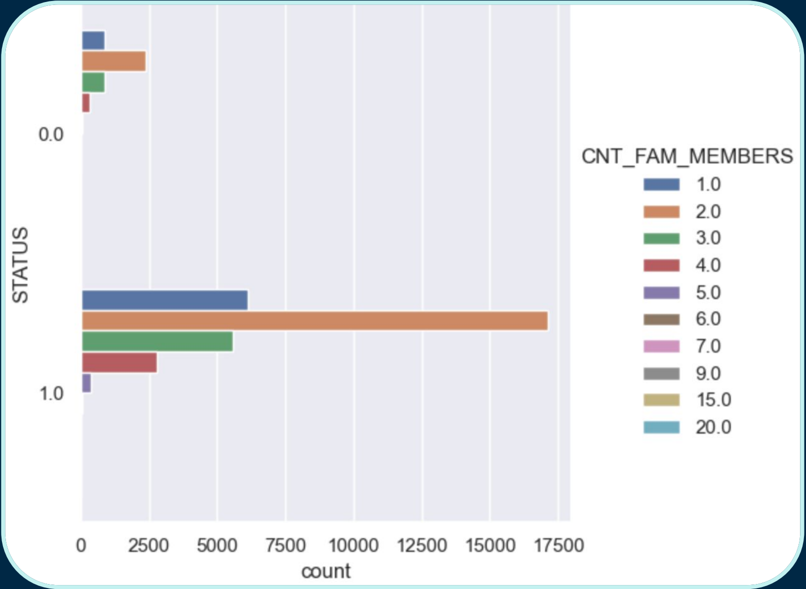
KEEP

- ❑ Total income
- ❑ Education type
- ❑ Marital status
- ❑ Employment status
- ❑ Occupation type
- ❑ Number of family members
- ❑ Gender
- ❑ Number of children
- ❑ Housing type
- ❑ Income type

REMOVE

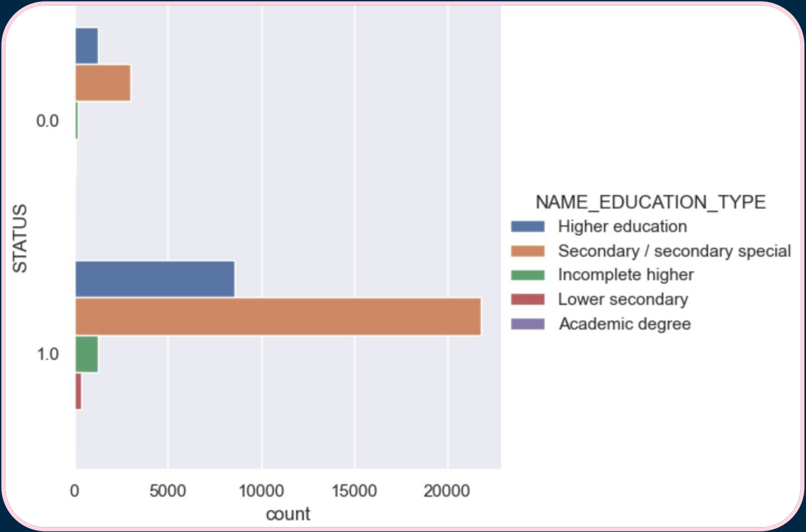
- ❑ Car ownership
- ❑ Property ownership
- ❑ Birthday
- ❑ Phone ownership
- ❑ Email ownership
- ❑ Mobile phone ownership
- ❑ Work phone ownership

CATEGORICAL PLOT



Number of
Family Members

Education Type



CHI-SQUARED TEST

```
 create a contingency table of counts
cont_table = pd.crosstab(merged_df["STATUS"], merged_df["OCCUPATION_TYPE"])

# perform chi-square test
chi2, p_val, dof, exp_freq = stats.chi2_contingency(cont_table)

# print results
print("Chi-square test results:")
print(f"Chi-square statistic: {chi2:.2f}")
print(f"P-value: {p_val:.4f}")
print(f"Degrees of freedom: {dof}")
print("Expected frequencies:")
print(exp_freq)
```

```
Chi-square test results:
Chi-square statistic: 41.85
P-value: 0.0012
Degrees of freedom: 18
Expected frequencies:
[[1.51648655e+02 6.73315138e+01 8.00401843e+01 4.38815728e+02
  2.61260938e+02 1.03868941e+01 1.69000878e+02 7.33192528e+00
  7.58976465e+02 2.13847821e+01 3.68062649e+02 1.47493897e+02
  1.38365650e+03 4.20363716e+01 9.65370162e+00 4.25862660e+02
  1.84520120e+01 7.23416628e+01 2.12625833e+01]
 [1.08935135e+03 4.83668486e+02 5.74959816e+02 3.15218427e+03
  1.87673906e+03 7.46131059e+01 1.21399912e+03 5.26680747e+01
  5.45202353e+03 1.53615218e+02 2.64393735e+03 1.05950610e+03
  9.93934350e+03 3.01963628e+02 6.93462984e+01 3.05913734e+03
  1.32547988e+02 5.19658337e+02 1.52737417e+02]]
```

Occupation Type

Total Income

```
# create a contingency table of counts
cont_table = pd.crosstab(merged_df["STATUS"], merged_df["Income Range"])

# perform chi-square test
chi2, p_val, dof, exp_freq = stats.chi2_contingency(cont_table)

# print results
print("Chi-square test results:")
print(f"Chi-square statistic: {chi2:.2f}")
print(f"P-value: {p_val:.4f}")
print(f"Degrees of freedom: {dof}")
print("Expected frequencies:")
print(exp_freq)
```

```
Chi-square test results:
Chi-square statistic: 19.48
P-value: 0.0016
Degrees of freedom: 5
Expected frequencies:
[[  38.84111806  576.56675248 1238.55976462  926.98381758  831.39352703
   335.65502023]
 [ 282.15888194 4188.43324752 8997.44023538 6734.01618242 6039.60647297
  2438.34497977]]
```

VARIABLE SELECTION 2

KEEP

- Total income
- Occupation type
- Education type
- Number of family members

REMOVE

- Gender
- Car ownership
- Property ownership
- Number of children
- Income type
- Marital status
- Housing type
- Birthday
- Employment status
- Phone ownership
- Email ownership
- Mobile phone ownership
- Work phone ownership

MACHINE LEARNING

04

TARGET ENCODING

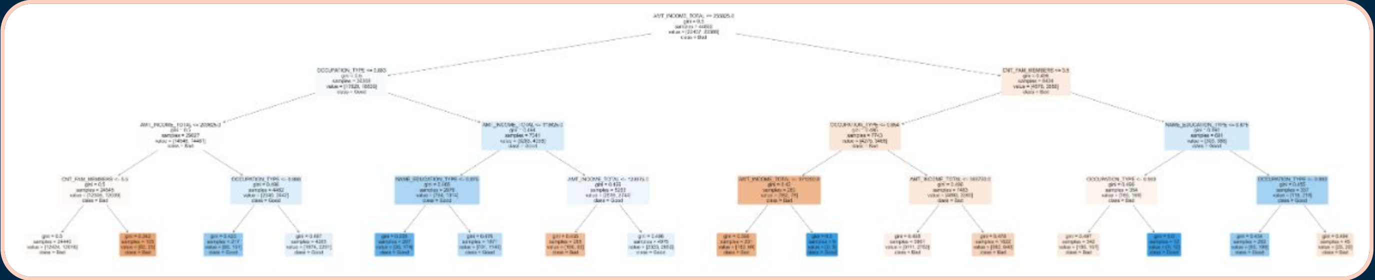
Used to convert categorical variables to numerical variables

```
# Use target encoder to encode the non-numerical columns
# Define the target encoder with smoothing to avoid overfitting
encoder = ce.TargetEncoder(cols=["NAME_INCOME_TYPE", "NAME_EDUCATION_TYPE", "NAME_FAMILY_STATUS", "NAME_HOUSING_TYPE",
                                 "OCCUPATION_TYPE","CODE_GENDER","FLAG_OWN_CAR","FLAG_OWN_REALTY"], smoothing=0.2)
print("Converted non-numerical columns to numerical columns using target encoder with smoothing")

# Fit and transform the encoder on the data
merged_df = encoder.fit_transform(merged_df, merged_df["STATUS"])

# Show number of numerical and non-numerical columns in the dataframe
print("\n[POST CONVERSION]\n Number of numerical columns: {}".format(merged_df.select_dtypes(include=np.number).shape[1]))
print("\n[POST CONVERSION]\n Number of non-numerical columns: {}".format(merged_df.select_dtypes(exclude=np.number).shape[1]))
```

```
Converted non-numerical columns to numerical columns using target encoder with smoothing

[POST CONVERSION]
 Number of numerical columns: 19

[POST CONVERSION]
 Number of non-numerical columns: 1
```

DECISION TREE

- Classify cases into groups
- Identify relationships between categories and variables
- Can show the relationships of several variables and STATUS



UPSCALING

- ❑ Initially had a high False Positive rate due to the data for SAMPLE being skewed

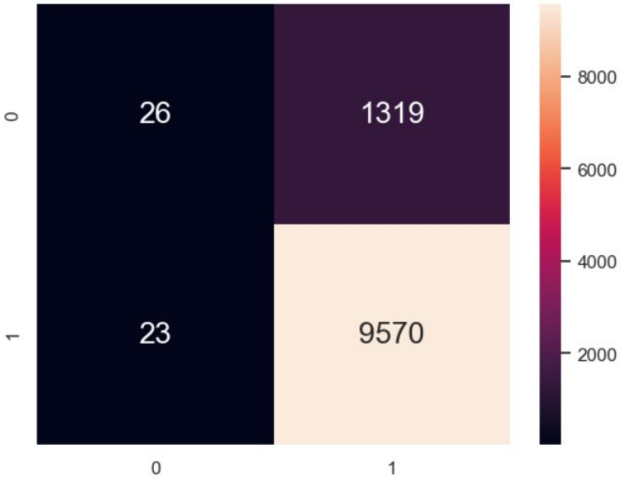
- ❑ To balance it, we upscaled the data

```python
# Upscale Bad to match Good
from sklearn.utils import resample

creditBad = merged_df[merged_df.STATUS == 1]
creditGood = merged_df[merged_df.STATUS == 0]

# Upscale the good samples
creditgood_up = resample(creditGood,
                         replace=True,
                         n_samples=creditBad.shape[0])

# Combine the two classes back after upsampling
scaled_df = pd.concat([creditBad, creditgood_up])

# Check the ratio of the classes
scaled_df['STATUS'].value_counts()
```

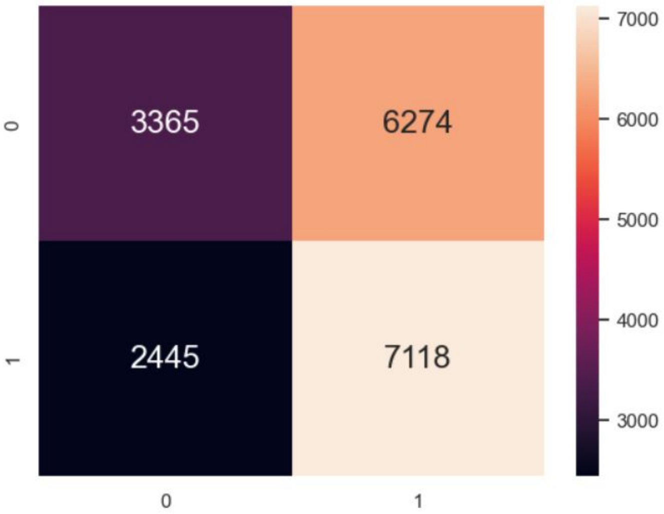
```
1.0    32002
0.0    32002
Name: STATUS, dtype: int64
```

UPSCALING

```
Test Data
Accuracy  :      0.8773084658987018

TPR Test :       0.9976024184301053
TNR Test :       0.019330855018587362

FPR Test :       0.9806691449814127
FNR Test :       0.002397581569894715
```

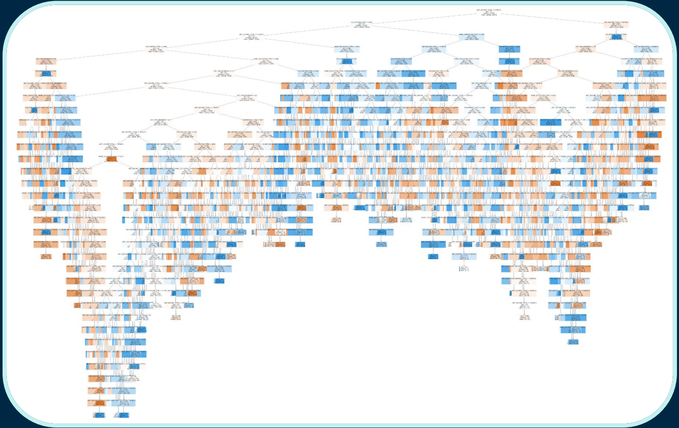



```
Test Data
Accuracy  :      0.5459327153421518

TPR Test :       0.7443270940081564
TNR Test :       0.3491026040045648

FPR Test :       0.6508973959954352
FNR Test :       0.25567290599184356
```



RANDOM FOREST

- Builds several decision trees using different samples from the data
- Takes the majority for classification, average for regression



GRIDSEARCH CROSS-VALIDATION

- Used to find the optimal values for hyperparameters
- Using hyperparameters best suited to the model helps to increase the model's accuracy

```
GridSearchCV(cv=5, estimator=RandomForestClassifier(),
             param_grid={'max_depth': array([14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24]),
                         'n_estimators': array([100, 200, 300, 400, 500])},
             scoring='balanced_accuracy')
```

```
RandomForestClassifier(max_depth=21, n_estimators=500)
0.5983409389822494
```

INSIGHTS

05

The variable that affects a person’s ability to pay off loans on time the most is

TOTAL INCOME.

NEW TOOLS AND TECHNIQUES

.MERGE

Used to merge two dataframes together

UPSCALE

Used to balance skewed data

MACHINE LEARNING

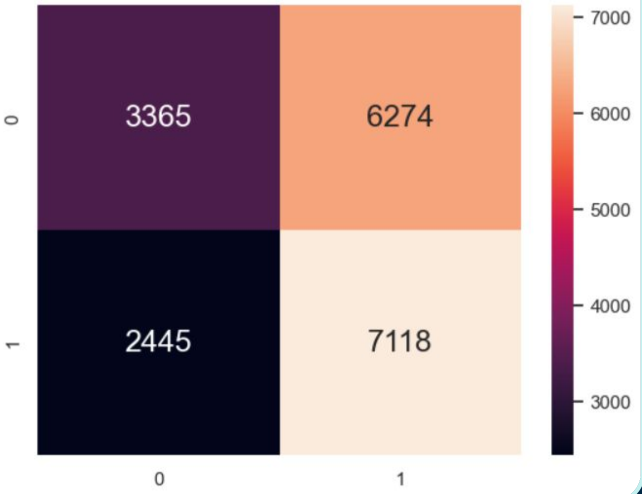
Random Forest, GridSearchCV

Best model for our problem:

Random Forest with GridSearchCV

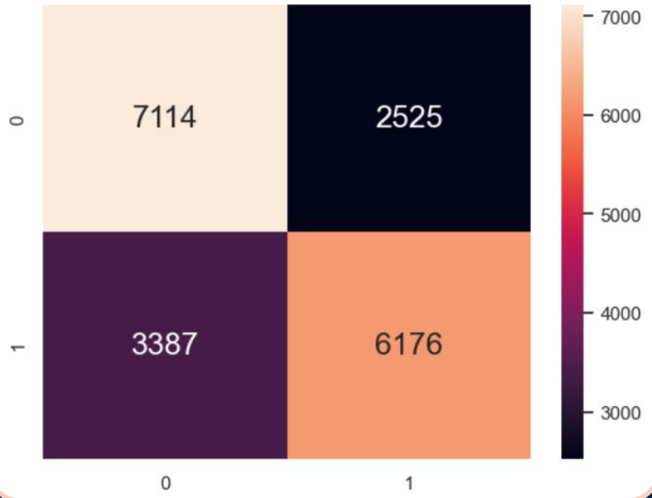
RECOMMENDATIONS

```
Test Data
Accuracy  :      0.5459327153421518

TPR Test :       0.7443270940081564
TNR Test :       0.3491026040045648

FPR Test :       0.6508973959954352
FNR Test :       0.25567290599184356
```



DECISION TREE

```
test Data
Accuracy  :      0.6921154046453495

TPR test :       0.6458224406566977
TNR test :       0.7380433654943459

FPR test :       0.2619566345056541
FNR test :       0.3541775593433023
```



RANDOM FOREST

RECOMMENDATIONS

DECISION TREE	Single model	Searches for the locally optimal solution at each step
RANDOM FOREST	Multiple models	More likely to find the globally optimal solution

RECOMMENDATIONS

USE MORE VARIABLES

Collecting data on other variables (eg: financial indicators) could improve accuracy

OTHER MACHINE LEARNING MODELS

Advanced models (eg: neural networks) may be better suited to solve this problem

THANK YOU!

REFERENCES

- Database: https://www.kaggle.com/datasets/rikdifos/credit-card-approval-prediction?select=credit_record.csv
- Decision Tree: https://www.ibm.com/docs/en/spss-statistics/25.0.0?topic=trees-creating-decision
- Random Forest: https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/
- GridSearchCV: https://www.mygreatlearning.com/blog/gridsearchcv/
- Comparing Decision Tree and Random Forest: https://vitalflux.com/differences-between-decision-tree-random-forest/#:~:text=Random%20forest%20is%20a%20ensemble,series%20of%20if%2Dthen%20rules.