Credit Card Approval Prediction using Machine Learning

Group 1

AIM

- Build a machine learning model to predict if an applicant is a “good” or “bad” client

- Data consists of two tables (Kaggle) in csv format:
    - Application record
    - Credit record

DATA LIMITATIONS

- Data Imbalance
  Data heavily imbalance towards “good credit”

```
full_df['STATUS'].value_counts()
```

```
0    28819
1     4291
Name: STATUS, dtype: int64
```

0 - Good Credit
1 - Bad Credit

- No “Target” column
  Binary values deduced using the credit record dataset

DATA PROCESSING & CLEANING

Credit Record Engineering

```
credit_record_df.head()
```

	CR_ID	ID	MONTHS_BALANCE	STATUS
0	0	5001711	0	X
1	1	5001711	-1	0
2	2	5001711	-2	0
3	3	5001711	-3	0
4	4	5001712	0	C

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: Overdue or bad debts, write-offs for more than 150 days

C: paid off that month

X: No loan for the month

- Dropped all X status
- C and 0 replaced with 0 (Good Credit)
- Remaining stati replaced with 1 (Bad Credit)

DATA PROCESSING & CLEANING

- Merged Credit Record & Application Record Datasets

	0	1	2	3	4
AR_ID	0	1	2	3	5
ID	5008804	5008805	5008806	5008808	5008810
CODE_GENDER	M	M	M	F	F
FLAG_OWN_CAR	Y	Y	Y	N	N
FLAG_OWN_REALTY	Y	Y	Y	Y	Y
CNT_CHILDREN	0	0	0	0	0
AMT_INCOME_TOTAL	427500.0	427500.0	112500.0	270000.0	270000.0
NAME_INCOME_TYPE	Working	Working	Working	Commercial associate	Commercial associate
NAME_EDUCATION_TYPE	Higher education	Higher education	Secondary / secondary special	Secondary / secondary special	Secondary / secondary special
NAME_FAMILY_STATUS	Civil marriage	Civil marriage	Married	Single / not married	Single / not married
NAME_HOUSING_TYPE	Rented apartment	Rented apartment	House / apartment	House / apartment	House / apartment
DAYS_BIRTH	-12005	-12005	-21474	-19110	-19110
DAYS_EMPLOYED	-4542	-4542	-1134	-3051	-3051
FLAG_MOBIL	1	1	1	1	1
FLAG_WORK_PHONE	1	1	0	0	0
FLAG_PHONE	0	0	0	1	1
FLAG_EMAIL	0	0	0	1	1
OCCUPATION_TYPE	Other	Other	Security staff	Sales staff	Sales staff
CNT_FAM_MEMBERS	2.0	2.0	2.0	1.0	1.0
STATUS	1	1	0	0	0

DATA PROCESSING & CLEANING

- Encoded Categorical Columns

	0	1	2	3	4
CNT_CHILDREN	0.0	0.0	0.0	0.0	0.0
AMT_INCOME_TOTAL	427500.0	427500.0	112500.0	270000.0	270000.0
NAME_INCOME_TYPE	2.0	2.0	2.0	2.0	2.0
NAME_EDUCATION_TYPE	3.0	3.0	1.0	1.0	1.0
NAME_FAMILY_STATUS	3.0	3.0	4.0	0.0	0.0
NAME_HOUSING_TYPE	1.0	1.0	2.0	2.0	2.0
DAYS_BIRTH	-12005.0	-12005.0	-21474.0	-19110.0	-19110.0
DAYS_EMPLOYED	-4542.0	-4542.0	-1134.0	-3051.0	-3051.0
FLAG_MOBIL	1.0	1.0	1.0	1.0	1.0
FLAG_WORK_PHONE	1.0	1.0	0.0	0.0	0.0
FLAG_PHONE	0.0	0.0	0.0	1.0	1.0
FLAG_EMAIL	0.0	0.0	0.0	1.0	1.0
OCCUPATION_TYPE	0.0	0.0	2.0	3.0	3.0
CNT_FAM_MEMBERS	2.0	2.0	2.0	1.0	1.0
CODE_GENDER_F	0.0	0.0	0.0	1.0	1.0
CODE_GENDER_M	1.0	1.0	1.0	0.0	0.0
FLAG_OWN_CAR_N	0.0	0.0	0.0	1.0	1.0
FLAG_OWN_CAR_Y	1.0	1.0	1.0	0.0	0.0
FLAG_OWN_REALTY_N	0.0	0.0	0.0	0.0	0.0
FLAG_OWN_REALTY_Y	1.0	1.0	1.0	1.0	1.0
STATUS	1.0	1.0	0.0	0.0	0.0

MACHINE LEARNING

Algorithms Used:

- Logistic Regression
- Random Forest Classifier
- Decision Tree Classifier
- Neural Networks

LOGISTIC REGRESSION (Unbalanced Data)

Model 1 output without class weights

Confusion Matrix

	Predicted 0	Predicted 1
Actual 0	7219	0
Actual 1	1059	0

Accuracy Score : 0.8720705484416525

Classification Report

	precision	recall	f1-score	support
0	0.87	1.00	0.93	7219
1	0.00	0.00	0.00	1059
accuracy			0.87	8278
macro avg	0.44	0.50	0.47	8278
weighted avg	0.76	0.87	0.81	8278

LOGISTIC REGRESSION (Balanced Data)

Model 1 output with class weights

Confusion Matrix

	Predicted 0	Predicted 1
Actual 0	4068	3151
Actual 1	531	528

Accuracy Score : 0.5552065716356608

Classification Report

	precision	recall	f1-score	support
0	0.88	0.56	0.69	7219
1	0.14	0.50	0.22	1059
accuracy			0.56	8278
macro avg	0.51	0.53	0.46	8278
weighted avg	0.79	0.56	0.63	8278

LOGISTIC REGRESSION (Balanced Data)

Model 1 output with Hyperparameter Tuning

Confusion Matrix

	Predicted 0	Predicted 1
Actual 0	4071	3148
Actual 1	526	533

Accuracy Score : 0.5561729886446002

	precision	recall	f1-score	support
0	0.89	0.56	0.69	7219
1	0.14	0.50	0.22	1059
accuracy			0.56	8278
macro avg	0.52	0.53	0.46	8278
weighted avg	0.79	0.56	0.63	8278

RANDOM FOREST CLASSIFIER (Unbalanced Data)

Confusion Matrix

	Predicted 0	Predicted 1
Actual 0	6958	286
Actual 1	692	342

```
Accuracy Score : 0.8818555206571635
Classification Report
              precision    recall  f1-score   support

           0       0.91      0.96      0.93      7244
           1       0.54      0.33      0.41      1034

    accuracy                           0.88      8278
   macro avg       0.73      0.65      0.67      8278
weighted avg       0.86      0.88      0.87      8278
```

RANDOM FOREST CLASSIFIER (Balanced Data)

```
Confusion Matrix
```

	Predicted 0	Predicted 1
Actual 0	6638	583
Actual 1	614	6575

```
Accuracy Score : 0.9169326856349758
Classification Report
              precision    recall  f1-score   support

           0       0.92      0.92      0.92      7221
           1       0.92      0.91      0.92      7189

    accuracy                           0.92     14410
   macro avg       0.92      0.92      0.92     14410
weighted avg       0.92      0.92      0.92     14410
```

DECISION TREE CLASSIFIER

(Training with non-optimized, non-resampled dataset)

Confusion Matrix

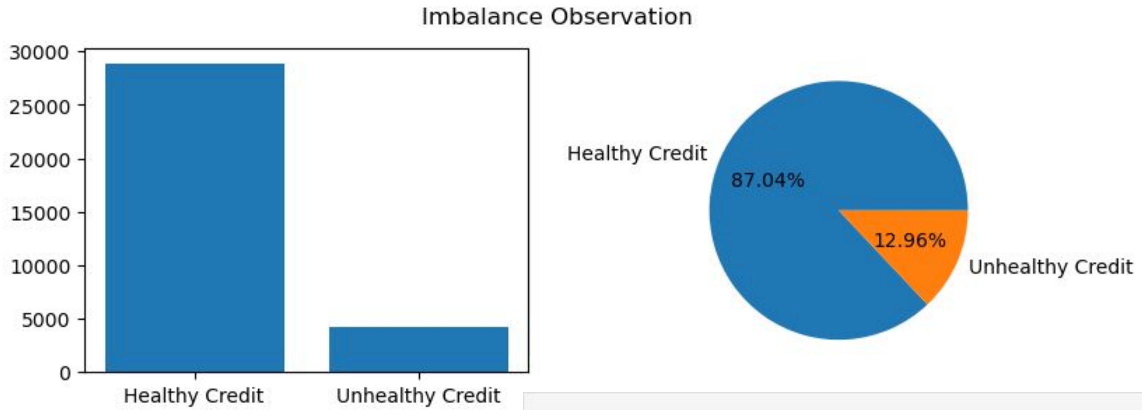
	Predicted 0	Predicted 1
Actual 0	6865	379
Actual 1	673	361

```
Accuracy Score : 0.8729161633244745
Classification Report
              precision    recall  f1-score   support

           0       0.91      0.95      0.93      7244
           1       0.49      0.35      0.41      1034

    accuracy                           0.87      8278
   macro avg       0.70      0.65      0.67      8278
weighted avg       0.86      0.87      0.86      8278
```

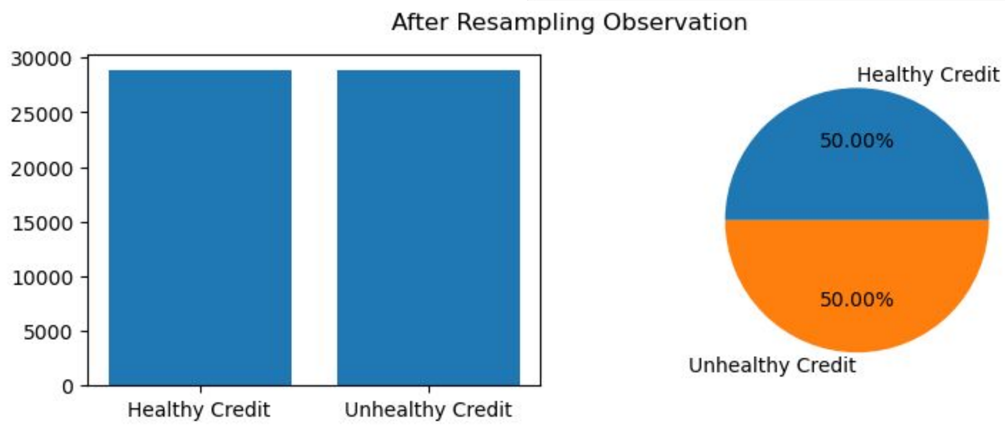
DECISION TREE CLASSIFIER
(Training with non-optimized, non-resampled dataset)




```python
# Fixing the imbalance
# summarize class distribution
# data = counter
data = {'Healthy Credit': list(counter.values())[1], 'Unhealthy Credit': list(counter.values())[0]}
names = list(data.keys())
values = list(data.values())

labels = names
sizes = values

fig, axs = plt.subplots(1,2, figsize=(9, 3))
axs[0].bar(names, values)
axs[1].pie(sizes, labels=names, autopct='%1.2f%%')
fig.suptitle('Imbalance Observation')
```



DECISION TREE CLASSIFIER
(Training with resampled dataset)

Confusion Matrix

	Predicted 0	Predicted 1
Actual 0	6531	690
Actual 1	769	6420

```
Accuracy Score : 0.8987508674531576
Classification Report
              precision    recall  f1-score   support

           0       0.89      0.90      0.90      7221
           1       0.90      0.89      0.90      7189

    accuracy                           0.90     14410
   macro avg       0.90      0.90      0.90     14410
weighted avg       0.90      0.90      0.90     14410
```

NEURAL NETWORKS
Pre Optimisation Values

```
# Define the model - deep neural net, i.e., the number of input features and hidden nodes for each layer.
number_input_features = len(X_train_scaled[0])
hidden_nodes_layer1 =  20
hidden_nodes_layer2 =  10
outer_layer = 1

# Define the deep learning model
nn_model = tf.keras.models.Sequential()

# First hidden layer
nn_model.add(tf.keras.layers.Dense(units=hidden_nodes_layer1, activation="relu", input_dim=number_input_fe

# Second hidden layer
nn_model.add(tf.keras.layers.Dense(units=hidden_nodes_layer2, activation="relu"))

# Output layer
nn_model.add(tf.keras.layers.Dense(units=outer_layer, activation="sigmoid"))

# Check the structure of the model
print(nn_model.summary())
```

NEURAL NETWORKS
Pre Optimisation Results

```python
# Compile the Sequential model together and customize metrics
nn_model.compile(loss="binary_crossentropy", optimizer="adam", metrics=["accuracy"])
```

```python
# Train the model
fit_model = nn_model.fit(X_train_scaled, y_train, epochs=50)
```

```python
# Evaluate the model using the test data
nn_model_loss, nn_model_accuracy = nn_model.evaluate(X_test_scaled, y_test, verbose=2)
print(f"Loss: {nn_model_loss}, Accuracy: {nn_model_accuracy}")
```

```
259/259 - 0s - loss: 0.3750 - accuracy: 0.8744 - 351ms/epoch - 1ms/step
Loss: 0.3749982416629791, Accuracy: 0.8743658065795898
```

NEURAL NETWORKS
Hyperparameter Tuning Results

```python
# Import the kerastuner library
import keras_tuner as kt

tuner = kt.Hyperband(
    create_model,
    objective="val_accuracy",
    max_epochs=20,
    hyperband_iterations=2)
```

```python
# Run the kerastuner search for best hyperparameters
tuner.search(X_train_scaled,y_train,epochs=20,validation_data=(X_test_scaled,y_test))
```

```
Trial 60 Complete [00h 01m 10s]
val_accuracy: 0.8741241693496704

Best val_accuracy So Far: 0.8750905990600586
Total elapsed time: 00h 17m 16s
INFO:tensorflow:Oracle triggered exit
```

NEURAL NETWORKS
Hyperparameter Tuning Results

```
# Get best model hyperparameters
best_hyper = tuner.get_best_hyperparameters(1)[0]
best_hyper.values
```

```
{'activation': 'tanh',
 'first_units': 9,
 'num_layers': 6,
 'units_0': 3,
 'units_1': 9,
 'units_2': 5,
 'units_3': 3,
 'units_4': 9,
 'units_5': 7,
 'tuner/epochs': 20,
 'tuner/initial_epoch': 0,
 'tuner/bracket': 0,
 'tuner/round': 0}
```

```
best_model = tuner.get_best_models(1)[0]
model_loss, model_accuracy = best_model.evaluate(X_test_scaled,y_test,verbose=2)
print(f"Loss: {model_loss}, Accuracy: {model_accuracy}")
```

```
259/259 - 2s - loss: 0.3752 - accuracy: 0.8751 - 2s/epoch - 6ms/step
Loss: 0.37516242265701294, Accuracy: 0.8750905990600586
```

SUMMARY

Model	Accuracy Before Optimisation	Accuracy After Optimisation
Logistic Regression	55.5%	55.6%
Random Forest	88.2%	91.7%
Decision Tree	87.3%	89.9%
Neural Networks	87.4%	87.5%

Based on the accuracy from all the models, the best model to use in this case is the Random Forest classifier.


BANK
4067


THANK YOU!