## Introduction

The problem at hand involves predicting the outcome of a banking institution's direct marketing campaigns based on phone calls. The goal is to determine whether a client would subscribe ('yes') or not subscribe ('no') to a bank term deposit. This task was approached using two different Machine Learning Algorithms: Decision Tree and Random Forest.

## Data Processing

The dataset used for this task consists of 41188 instances with 21 attributes, including both numerical and categorical data. The attributes range from personal information about the clients (like 'age', 'job', 'marital', 'education', 'default', 'housing', 'loan') to information related to the last contact of the current campaign (like 'contact', 'month', 'day_of_week', 'duration') and other attributes (like 'campaign', 'pdays', 'previous', 'poutcome', 'emp.var.rate', 'cons.price.idx', 'cons.conf.idx', 'euribor3m', 'nr.employed').

## The data preprocessing involved several steps:

1. **Encoding categorical features**: The categorical features were transformed into numerical values using LabelEncoder. This is necessary because machine learning algorithms work better with numerical data.

```
le = LabelEncoder()

categorical_features = ['job', 'marital', 'education', 'default', 'housing', 'loan', 'contact', 'month', 'day_of_week', 'poutcome']

for feature in categorical_features:

 df[feature] = le.fit_transform(df[feature])
```

**1. Feature scaling**: The features were scaled using StandardScaler to ensure that all features contribute equally to the model performance.

```
scaler = StandardScaler()

X_scaled = scaler.fit_transform(X)
```

**1. Dimensionality reduction**: Principal Component Analysis (PCA) was applied to reduce the dimensionality of the dataset. This helps in reducing the computational complexity of the model.

```
pca = PCA(n_components=2)

X_pca = pca.fit_transform(X_scaled)
```

**1. Splitting the dataset**: The dataset was split into training and testing sets, with 80% of the data used for training the models and 20% used for testing their performance.

```
X_train, X_test, y_train, y_test = train_test_split(X_pca, y, test_size=0.2, random_state=42)
```

## Modelling

Two machine learning algorithms were chosen for this task: Decision Tree and Random Forest. These algorithms were chosen because they

work well with both numerical and categorical data, and they are interpretable, which is important in a business context.

The models were trained on the preprocessed training data. The performance of the models was then evaluated on the testing data.

dt = DecisionTreeClassifier()

dt.fit(X_train,y_train)

rf = RandomForestClassifier()

rf.fit(X_train,y_train)

## Model Evaluation

The performance of the models was evaluated using several metrics, including precision, recall, f1-score, and support for both 'no' and

'yes' classes, as well as accuracy, macro average, and weighted average.

For the Decision Tree model, the precision, recall, and f1-score for the 'no' class were all 0.91, and for the 'yes' class, these metrics were all 0.32.

For the Random Forest model, the precision, recall, and f1-score for the 'no' class were 0.91, 0.96, and 0.94 respectively, and for the 'yes' class, these metrics were 0.67, 0.34, and 0.45 respectively. The overall accuracy of the model was 0.88.

Confusion matrices were also computed for both models to provide a summary of prediction results on the classification problem. The number of correct and incorrect predictions were summarized with count values and broken down by each class.

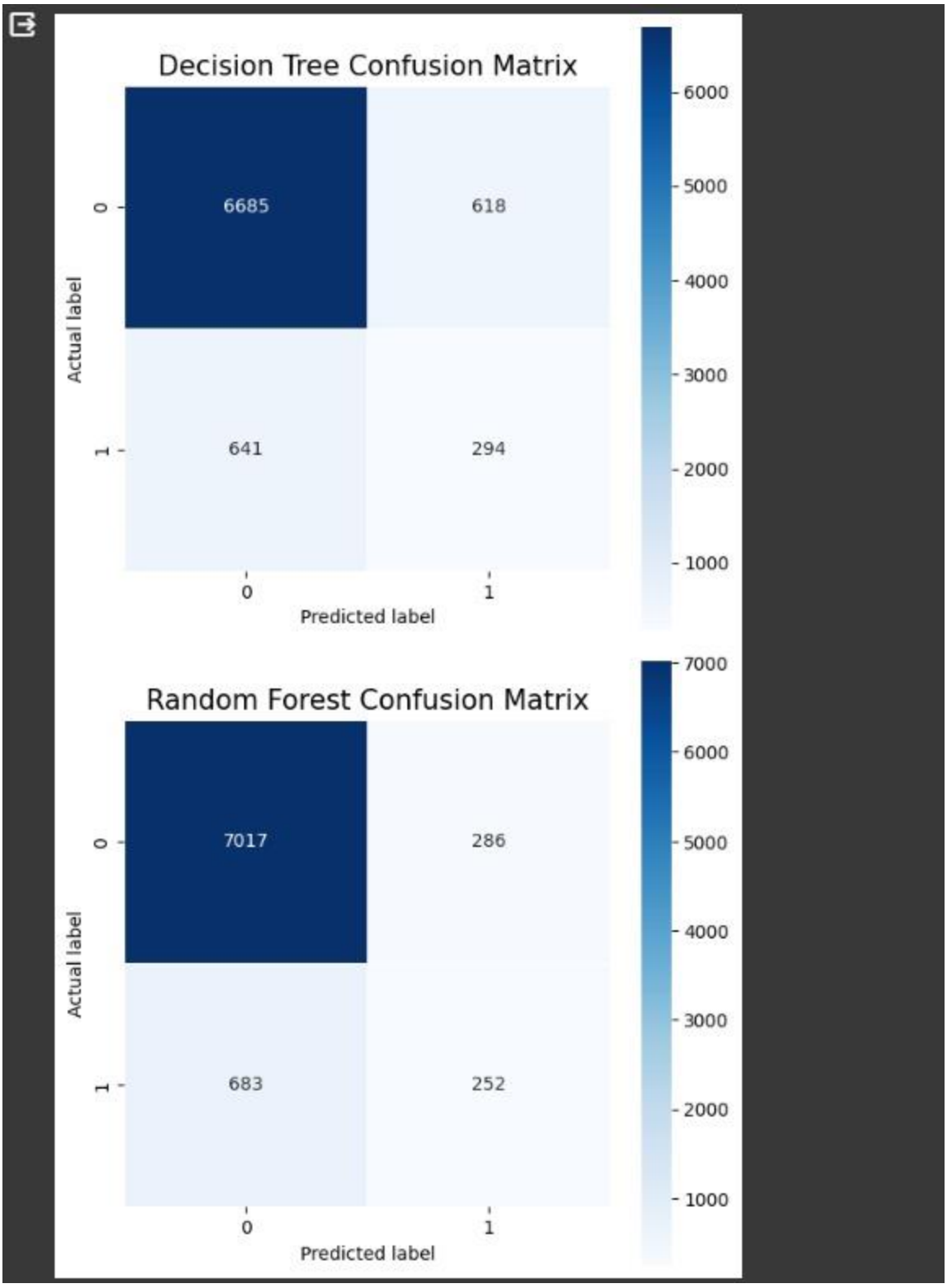
```python
from sklearn.metrics import confusion_matrix
import seaborn as sns

# Compute confusion matrix for Decision Tree
dt_cm = confusion_matrix(y_test, dt_predictions)

# Compute confusion matrix for Random Forest
rf_cm = confusion_matrix(y_test, rf_predictions)

# Display the confusion matrix for Decision Tree
plt.figure(figsize=(6,6))
sns.heatmap(dt_cm, annot=True, fmt=".0f", linewidths=.5, square = True, cmap = 'Blues');
plt.ylabel('Actual label');
plt.xlabel('Predicted label');
plt.title('Decision Tree Confusion Matrix', size = 15);
plt.show()

# Display the confusion matrix for Random Forest
plt.figure(figsize=(6,6))
sns.heatmap(rf_cm, annot=True, fmt=".0f", linewidths=.5, square = True, cmap = 'Blues');
plt.ylabel('Actual label');
plt.xlabel('Predicted label');
plt.title('Random Forest Confusion Matrix', size = 15);
plt.show()
```


Decision Tree Confusion Matrix
6685
618
641
294
Actual label
Predicted label
0
1
6000
5000
4000
3000
2000
1000
Random Forest Confusion Matrix
7017
286
683
252
Actual label
Predicted label
0
1
7000
6000
5000
4000
3000
2000
1000

## Conclusion

The task of predicting the outcome of a banking institution's direct marketing campaigns based on phone calls was successfully approached using Decision Tree and Random Forest algorithms. The models were trained and evaluated on a dataset consisting of both numerical and categorical data. The performance of the models was evaluated using several metrics, and confusion matrices were computed to provide a summary of prediction results.

The main scientific bottleneck in this task was dealing with both numerical and categorical data in the dataset. This was overcome by encoding the categorical features into numerical values and scaling all features to ensure they contribute equally to the model performance.

Comparing the results from the Decision Tree and Random Forest models, it can be seen that the Random Forest model performed better in terms of precision, recall, and f1-score for the 'yes' class, and overall accuracy. This could be due to the fact that Random Forest is an ensemble method that combines multiple decision trees to make a more accurate prediction.

In conclusion, this task demonstrated the effectiveness of Decision Tree and Random Forest algorithms in predicting the outcome of a banking institution's direct marketing campaigns based on phone calls. Future work could explore the use of other machine learning algorithms and feature selection methods to further improve the prediction accuracy.