Technical Report: Direct Marketing Dataset Analysis and Decision Tree Classification

Objective:

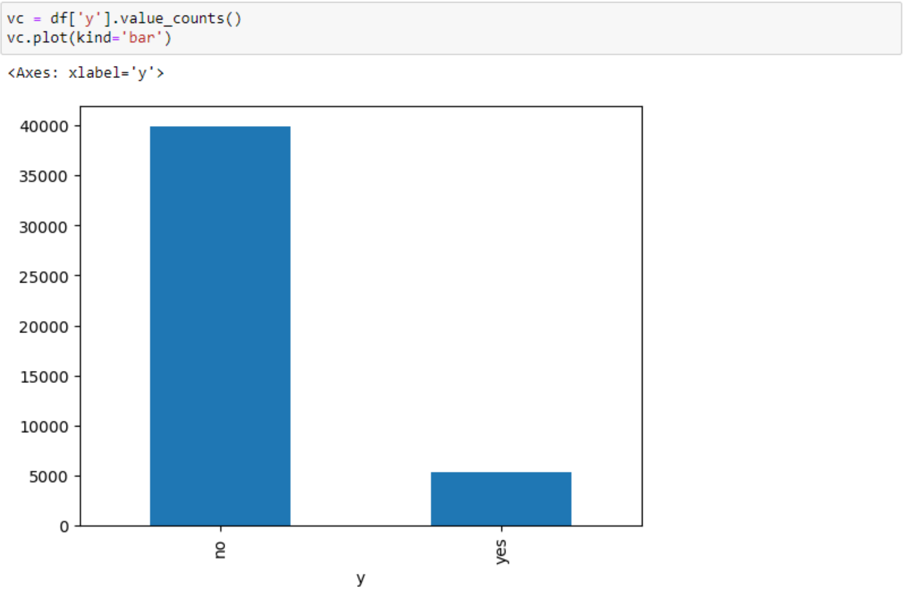
The objective of this technical report is to analyze the Direct Marketing Dataset and develop a Decision Tree Classifier model to predict customer response to marketing campaigns.

Data Preprocessing:

- Data Loading: The dataset was loaded from the 'direct_marketing_dataset.csv' file using Pandas.

Exploratory Data Analysis (EDA):

- Initial exploration included examining the first 5 rows of the dataset to understand its structure.
- Visualizing the distribution of the target variable ('y') using a bar plot to assess class imbalance.

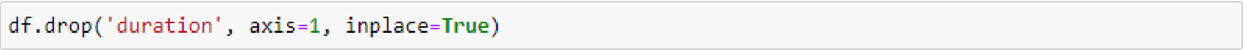


```
vc = df['y'].value_counts()
vc.plot(kind='bar')
```

```
<Axes: xlabel='y'>
```

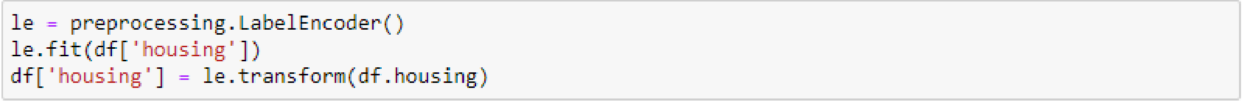
Data Preprocessing:

- Dropping the 'duration' column to avoid data leakage.

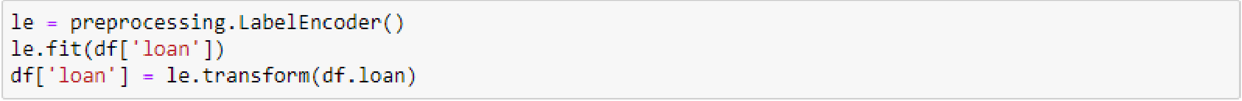


```
df.drop('duration', axis=1, inplace=True)
```

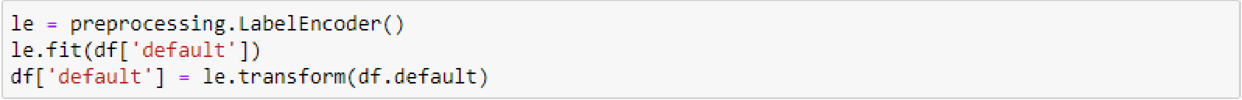
- Encoding categorical variables using LabelEncoder and OrdinalEncoder.
- One-hot encoding categorical variables with multiple categories.



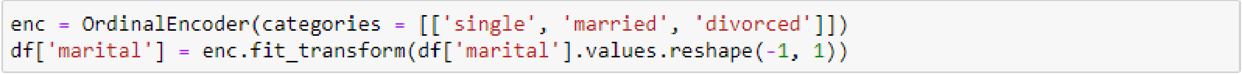
```
le = preprocessing.LabelEncoder()
le.fit(df['housing'])
df['housing'] = le.transform(df.housing)
```



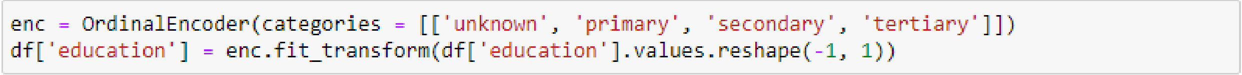
```
le = preprocessing.LabelEncoder()
le.fit(df['loan'])
df['loan'] = le.transform(df.loan)
```



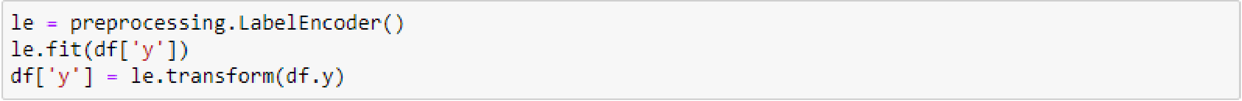
```
le = preprocessing.LabelEncoder()
le.fit(df['default'])
df['default'] = le.transform(df.default)
```



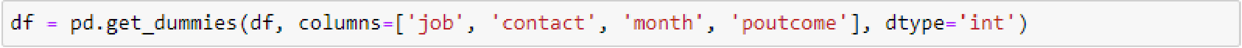
```
enc = OrdinalEncoder(categories = [['single', 'married', 'divorced']])
df['marital'] = enc.fit_transform(df['marital'].values.reshape(-1, 1))
```



```
enc = OrdinalEncoder(categories = [['unknown', 'primary', 'secondary', 'tertiary']])
df['education'] = enc.fit_transform(df['education'].values.reshape(-1, 1))
```

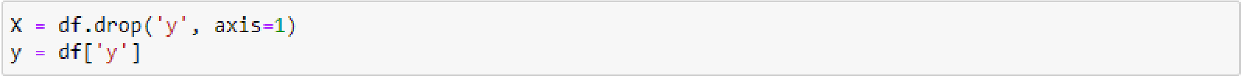


```
le = preprocessing.LabelEncoder()
le.fit(df['y'])
df['y'] = le.transform(df.y)
```

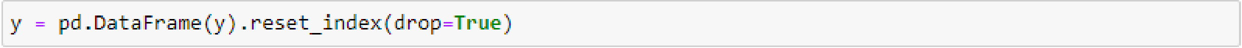


```
df = pd.get_dummies(df, columns=['job', 'contact', 'month', 'poutcome'], dtype='int')
```

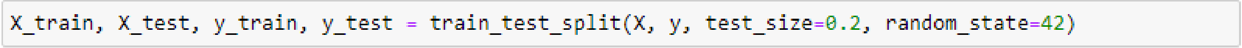
- Splitting the dataset into features (X) and the target variable (y).
- Splitting the data into training and testing sets with a ratio of 80:20.



```
X = df.drop('y', axis=1)
y = df['y']
```



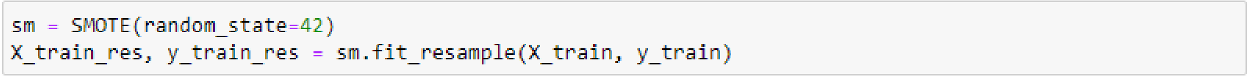
```
y = pd.DataFrame(y).reset_index(drop=True)
```



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

Handling Imbalanced Data:

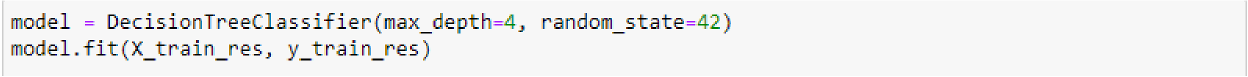
- Applied Synthetic Minority Over-sampling Technique (SMOTE) to balance the class distribution in the training set



```
sm = SMOTE(random_state=42)
X_train_res, y_train_res = sm.fit_resample(X_train, y_train)
```

Model Development:

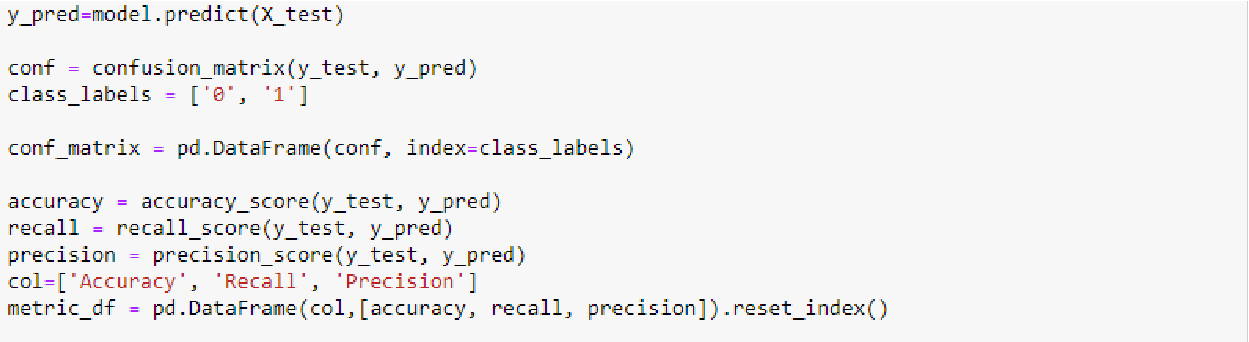
- Initialized a Decision Tree Classifier with a maximum depth of 4 and a random state of 42.
- Trained the model on the balanced training data (X_train_res, y_train_res).



```
model = DecisionTreeClassifier(max_depth=4, random_state=42)
model.fit(X_train_res, y_train_res)
```

Model Evaluation:

- Utilized the trained model to predict the target variable on the test set (X_test).
- Constructed a confusion matrix to evaluate classification performance.
- Computed accuracy, recall, and precision scores to assess model effectiveness.



```
y_pred=model.predict(X_test)

conf = confusion_matrix(y_test, y_pred)
class_labels = ['0', '1']

conf_matrix = pd.DataFrame(conf, index=class_labels)

accuracy = accuracy_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
col=['Accuracy', 'Recall', 'Precision']
metric_df = pd.DataFrame(col,[accuracy, recall, precision]).reset_index()
```

Conclusion:

The Decision Tree Classifier model achieved 0.80 accuracy, 0.51 recall, and 0.31 precision on the test dataset. These results indicate that the Decision Tree Classifier achieved an accuracy of 0.8, suggesting an overall correct classification rate of 80%. However, its recall is 0.51, indicating that only 51% of actual positive cases were correctly identified. Moreover, its precision is 0.31, implying that among instances predicted as positive, only 31% were truly positive. This suggests a need for improving the model's ability to correctly identify positive cases while reducing false positives.

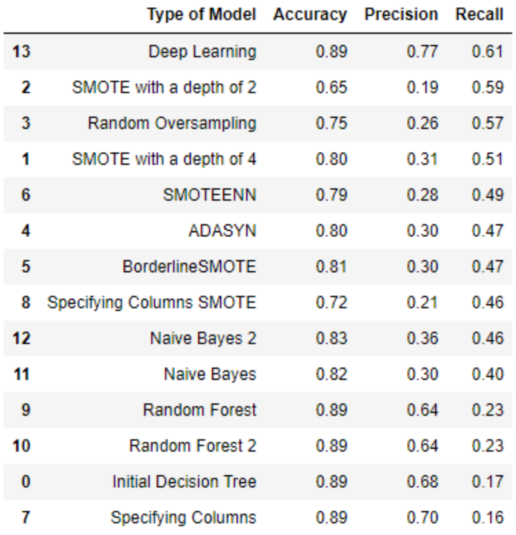
Recommendations:

Further experimentation with hyperparameters may improve model performance.

Evaluate the impact of feature engineering techniques on model performance.

Progress forward using deep learning techniques, such as Feedforward Neural Network, as this yielded the best result. The table on the left displays the outputs for all the models used.

This report summarizes the analysis and modeling process conducted on the Direct Marketing Dataset, providing insights into customer response prediction for marketing campaigns.



	Type of Model	Accuracy	Precision	Recall
13	Deep Learning	0.89	0.77	0.61
2	SMOTE with a depth of 2	0.65	0.19	0.59
3	Random Oversampling	0.75	0.26	0.57
1	SMOTE with a depth of 4	0.80	0.31	0.51
6	SMOTEENN	0.79	0.28	0.49
4	ADASYN	0.80	0.30	0.47
5	BorderlineSMOTE	0.81	0.30	0.47
8	Specifying Columns SMOTE	0.72	0.21	0.46
12	Naive Bayes 2	0.83	0.36	0.46
11	Naive Bayes	0.82	0.30	0.40
9	Random Forest	0.89	0.64	0.23
10	Random Forest 2	0.89	0.64	0.23
0	Initial Decision Tree	0.89	0.68	0.17
7	Specifying Columns	0.89	0.70	0.16