Data Glacier

Your Deep Learning Partner

Project: Bank Campaign

Data Glacier Virtual Internship

By: Riwaj Neupane

16th October 2023

Agenda

- Problem Statement
- Dataset information
- Problems in Data
- Approaches to solve problems
- EDA
- Findings
- Model Recommendation

Data Glacier
Your Deep Learning Partner

Problem Statement

Problem Description:

ABC Bank wants to sell its term deposit product to customers and before launching the product they want to develop a model which helps them in understanding whether a particular customer will buy their product or not (based on customer's past interaction with bank or other Financial Institution).

DataSet Information

Data Description:

The dataset going to be used for the analysis is called “bank-additional-full.csv”, which contains 41188 observations and 21 features, encompassing features related to clients’ basic information such as age, job, marital status, education, credit in default, housing, and loan; details about contact such as contact communication type, last contact month, last contact day, last contact duration, number of contacts, etc., and information about marketing campaigns like outcome, employment variation rate, consumer price index, consumer confidence index, euribor 3 month rate, and number of employees. We also have the target variable y, which is the answer for the yes-no question “has the client subscribed a term deposit?”, and it will be used in future prediction.

DataSet Information (contd...)

Feature Name	Type	Data Type	Number of unknowns	Number of Outliers	Comments
age	Numerical	int	0	381	Replace with upper bound defined as Q3+IQR
job	Categorical	str	330	0	Replace with mode
martial	Categorical	str	80	0	Replace with mode
education	Categorical	str	1731	0	
default	Categorical	str	8597	0	Leave unknown as its own type
housing	Categorical	str	990	0	Replace with mode
loan	Categorical	str	990	0	Replace with mode
contact	Categorical	str	0	0	
month	Categorical	str	0	0	
day_of_week	Categorical	str	0	0	
duration	Numerical	int	0	861	Replace with upper bound defined as Q3+IQR
campaign	Numerical	int	0	0	
previous	Numerical	int	0	0	
poutcome	Categorical	str	0	0	
emp.var.rate	Numerical	float64	0	0	
cons.price.idx	Numerical	float64	0	0	
cons.conf.idx	Numerical	float64	0	0	
euribor3m	Numerical	float64	0	0	
nr.employed	Numerical	float64	0	0	

Data Glacier
Your Deep Learning Partner

Problems in Data

There are 6 categorical features with missing data (job, education, marital, default, housing, & loan). There is one numerical feature (“duration”) that contains outlier data. Specifically, we have the mean for “duration” is around 258, but the maximum value is 4918, which indicates the existence of outliers. And in general, the dataset is imbalanced, as the target variable for the predictive classification model skews highly to the “N” case.

Data Glacier

Your Deep Learning Partner

Approaches to solve problems

For categorical data feature with unknown as category for (job, education, marital, default, housing, & loan). Replacing unknown we can use 2 method:

1. Replacing with mode:

   Example:

```
most_frequent_category = data['job'].mode()
data['job'] = data['job'].replace('unknown', most_frequent_category)
```

2. Replacing using RandomForestClassifier:

   Example:

Approaches to solve problems

Replacing using RandomForestClassifier:

Example:

```
# Separate known and unknown data
df_known = df[df['loan'] != 'unknown']
df_unknown = df[df['loan'] == 'unknown']

# Define categorical and numerical features
cat_features = ['job', 'marital', 'education', 'housing', 'loan', 'contact', 'month', 'day_of_week', 'poutcome']
num_features = ['age', 'duration', 'campaign', 'pdays', 'previous', 'emp.var.rate', 'cons.price.idx', 'cons.conf.idx', 'euribor3m

# One-hot encoding for categorical features
encoder = OneHotEncoder(handle_unknown='ignore')
encoder.fit(df_known[cat_features])
X_train_known_cat = encoder.transform(df_known[cat_features])
X_unknown_cat = encoder.transform(df_unknown[cat_features])

# Combine encoded categorical features with numerical features
X_train_known = sp.hstack((X_train_known_cat, df_known[num_features].values))
X_unknown = sp.hstack((X_unknown_cat, df_unknown[num_features].values))

# Target variable (categorical)
y_train = df_known['housing']

# Train a RandomForestClassifier
clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(X_train_known, y_train)

# Predict the unknown values
y_unknown = clf.predict(X_unknown)

# Replace the "unknown" values in the original DataFrame with predictions
df.loc[df['housing'] == 'unknown', 'housing'] = y_unknown
```

Data Glacier

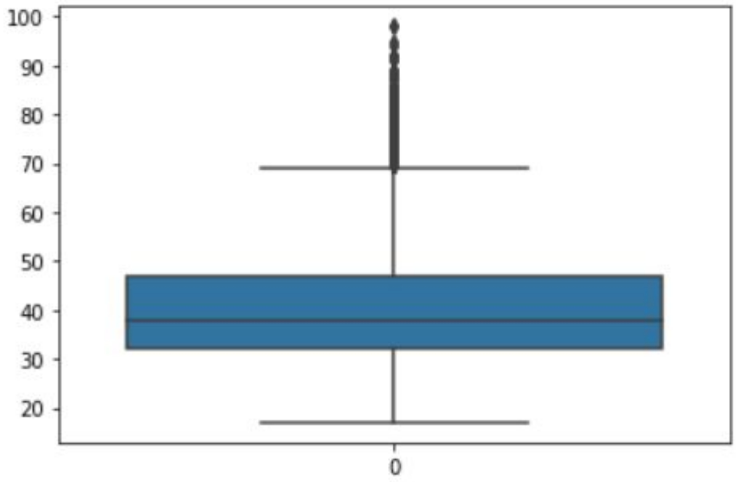
Your Deep Learning Partner

Approaches to solve problems

Detecting and Removing Outliers:

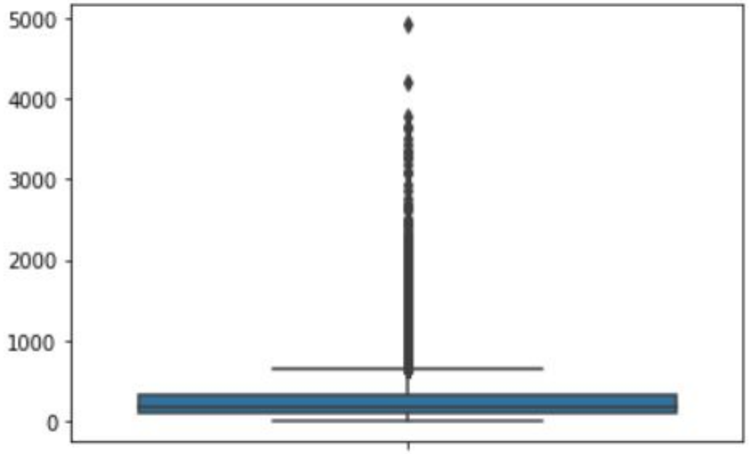
```
sns.boxplot(data.age)
```

```
<Axes: >
```




```
sns.boxplot(data.duration)
```

```
<Axes: >
```



Data Glacier

Your Deep Learning Partner

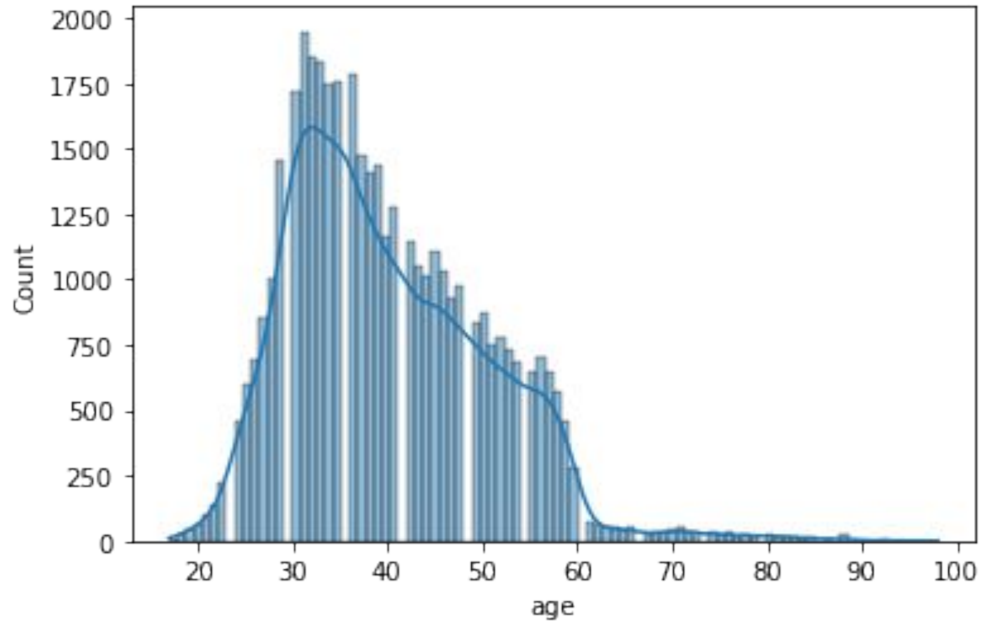
Approaches to solve problems

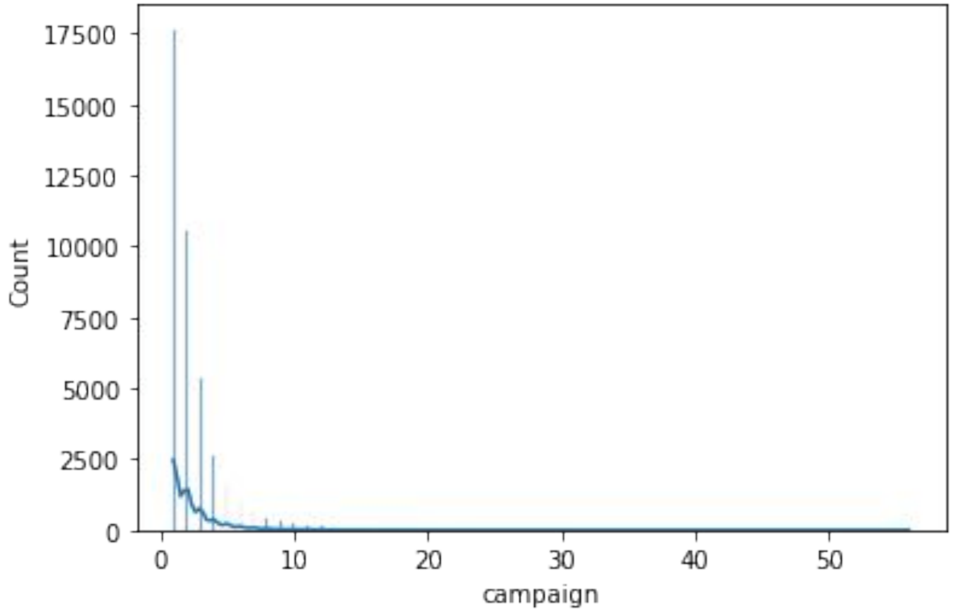
Detecting and Removing Outliers:

```python
def remove_outliers_iqr(df):
    outliers = {}
    for col in df.columns:
        if np.issubdtype(df[col].dtype, np.number):  # Check if the column contains numerical data
            # Calculate the IQR (Interquartile Range) for the column
            Q1 = df[col].quantile(0.25)
            Q3 = df[col].quantile(0.75)
            IQR = Q3 - Q1

            # Define lower and upper bounds for outliers
            lower_bound = Q1 - 1.5 * IQR
            upper_bound = Q3 + 1.5 * IQR

            # Find the indices of outliers
            outlier_indices = (df[col] < lower_bound) | (df[col] > upper_bound)

            # Create a DataFrame containing the outliers
            col_outliers = df[outlier_indices]

            # Add the outliers to the dictionary
            outliers[col] = col_outliers

    # Remove the rows containing outliers from the original DataFrame
    for col, col_outliers in outliers.items():
        df = df[~df.index.isin(col_outliers.index)]

    # Print information about removed outliers and the new shape
    if outliers:
        print('These outliers have been removed from your dataset:')
        for col, col_outliers in outliers.items():
            print(f'\nOutliers in column "{col}":')
            print(col_outliers)
    else:
        print('No outliers were found in the dataset.')

    print('\nNew shape is:', df.shape)
    return df
```

EDA

Distribution of Numerical Data



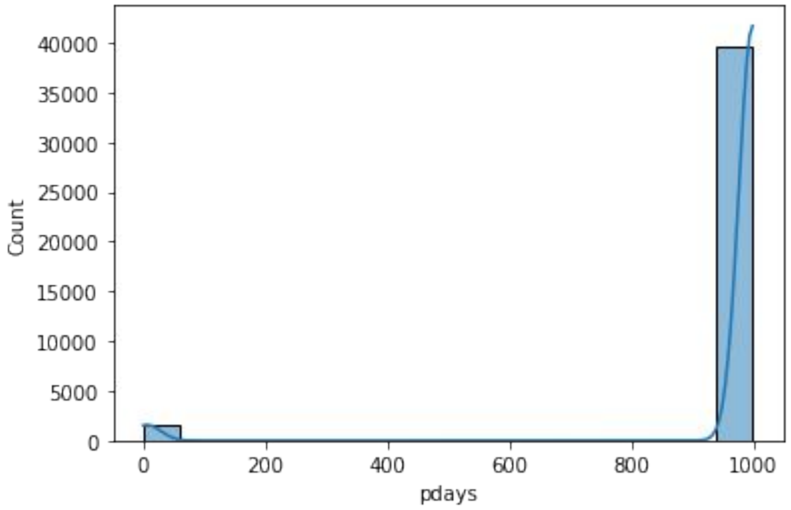


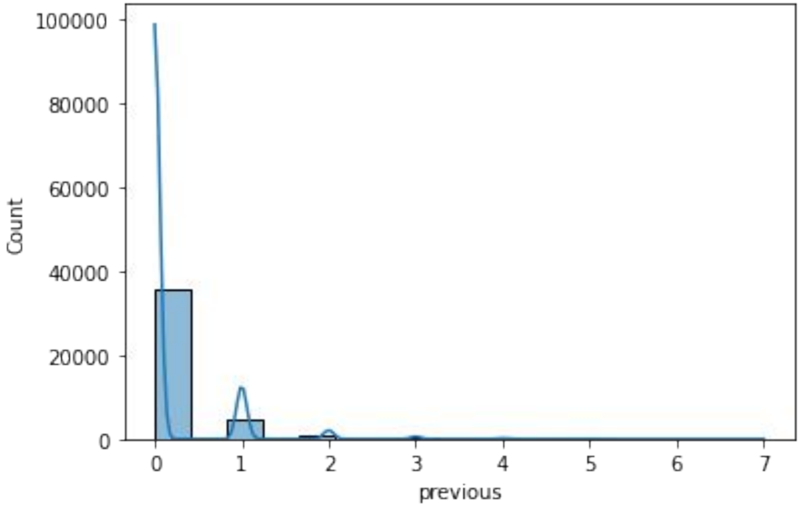
Data Glacier

Your Deep Learning Partner

EDA

Distribution of Numerical Data

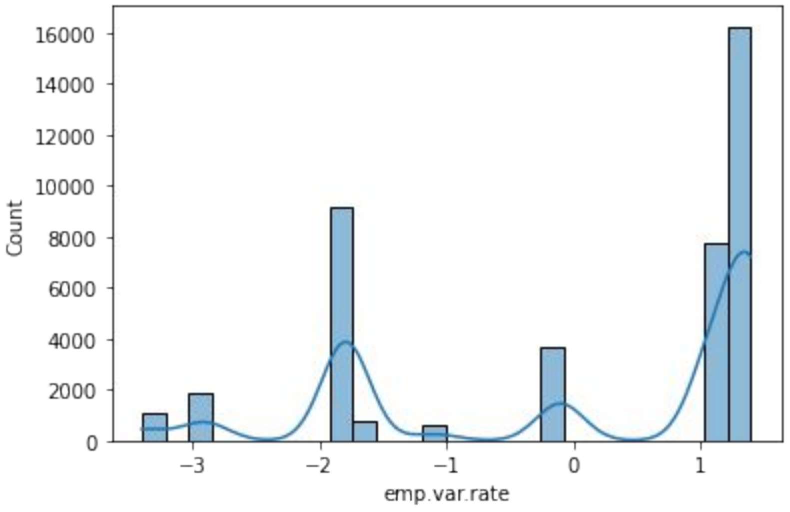


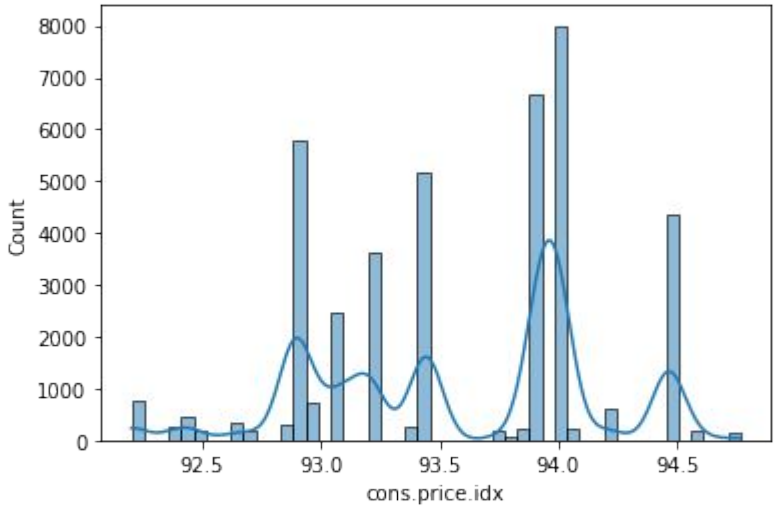


Data Glacier
Your Deep Learning Partner

EDA

Distribution of Numerical Data

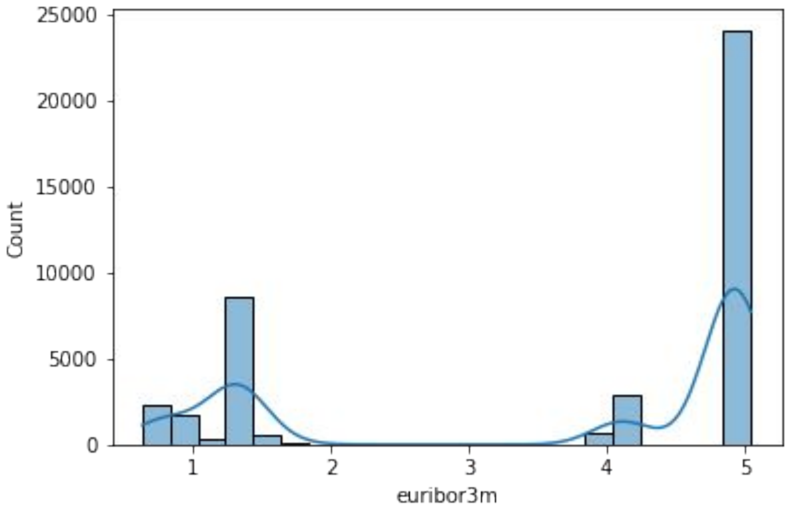


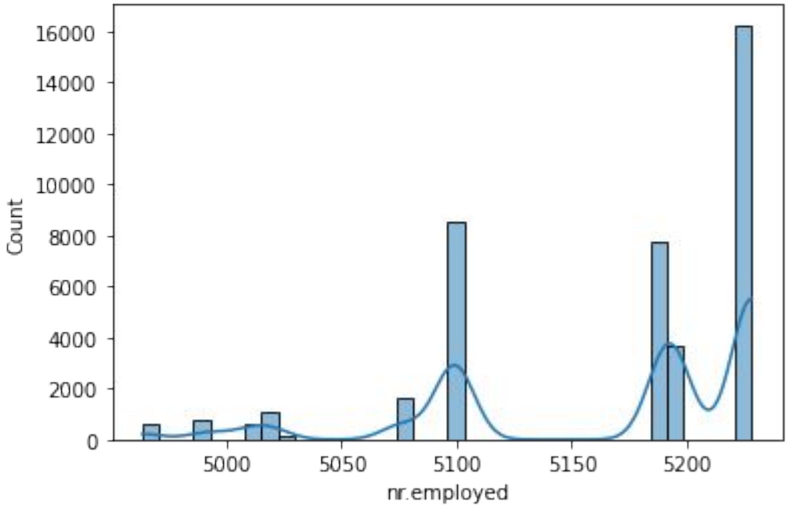


Data Glacier
Your Deep Learning Partner

EDA

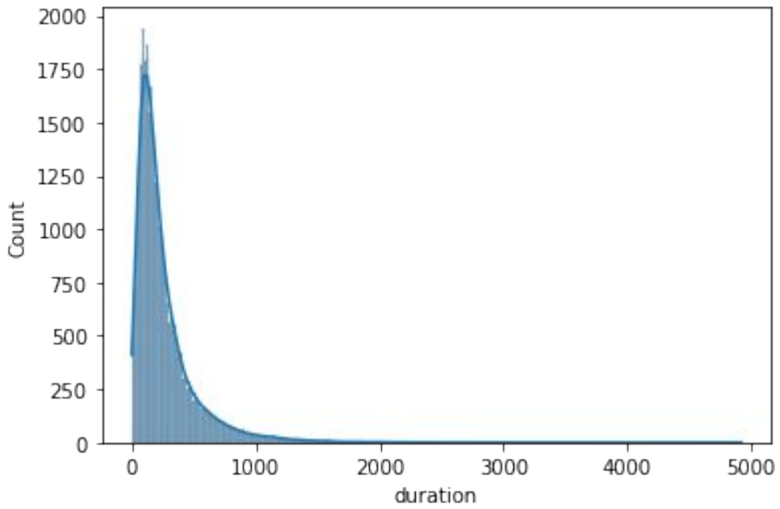
Distribution of Numerical Data





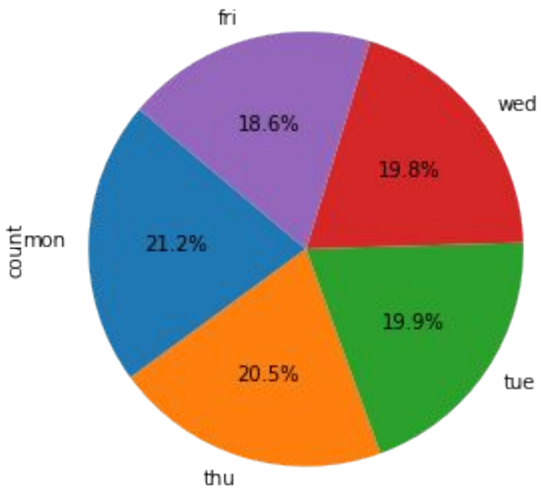
EDA

Distribution of Numerical Data

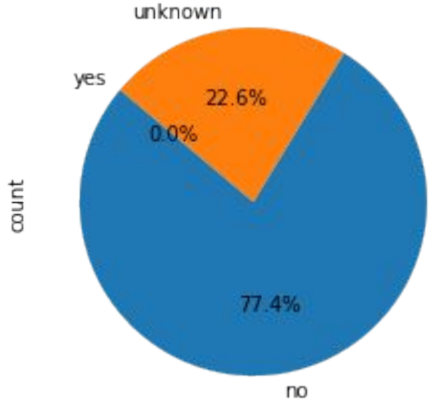


EDA

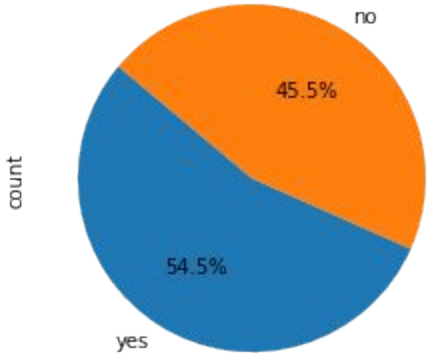
Distribution of Categorical Data



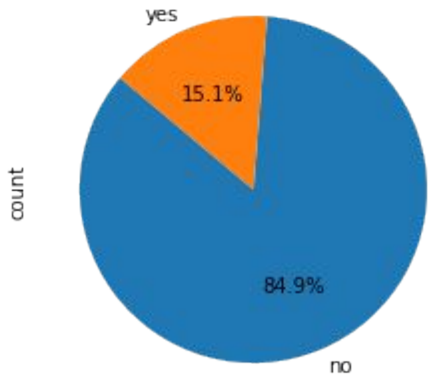
day_of_week



default



Housing

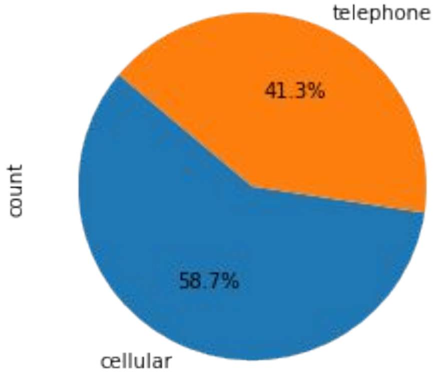


Loan

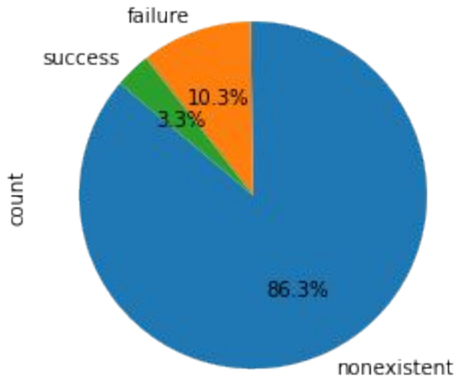
Data Glacier
Your Deep Learning Partner

EDA

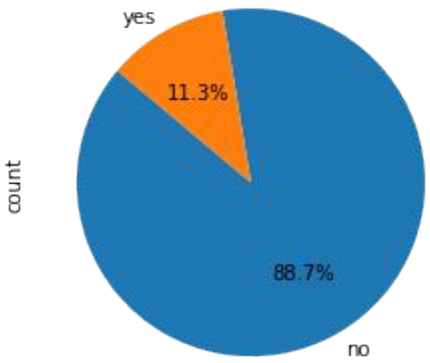
Distribution of Categorical Data



Telephone



poutcome



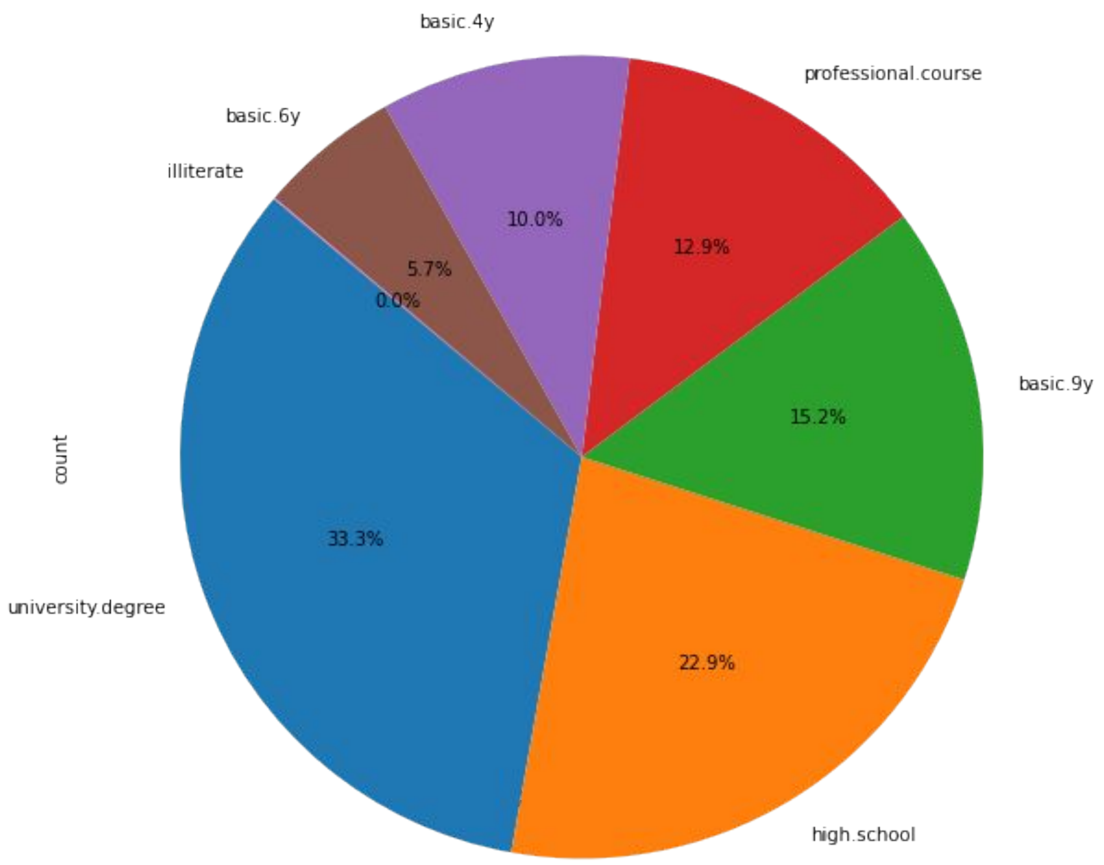
y

Data Glacier

Your Deep Learning Partner

EDA

Distribution of Categorical Data

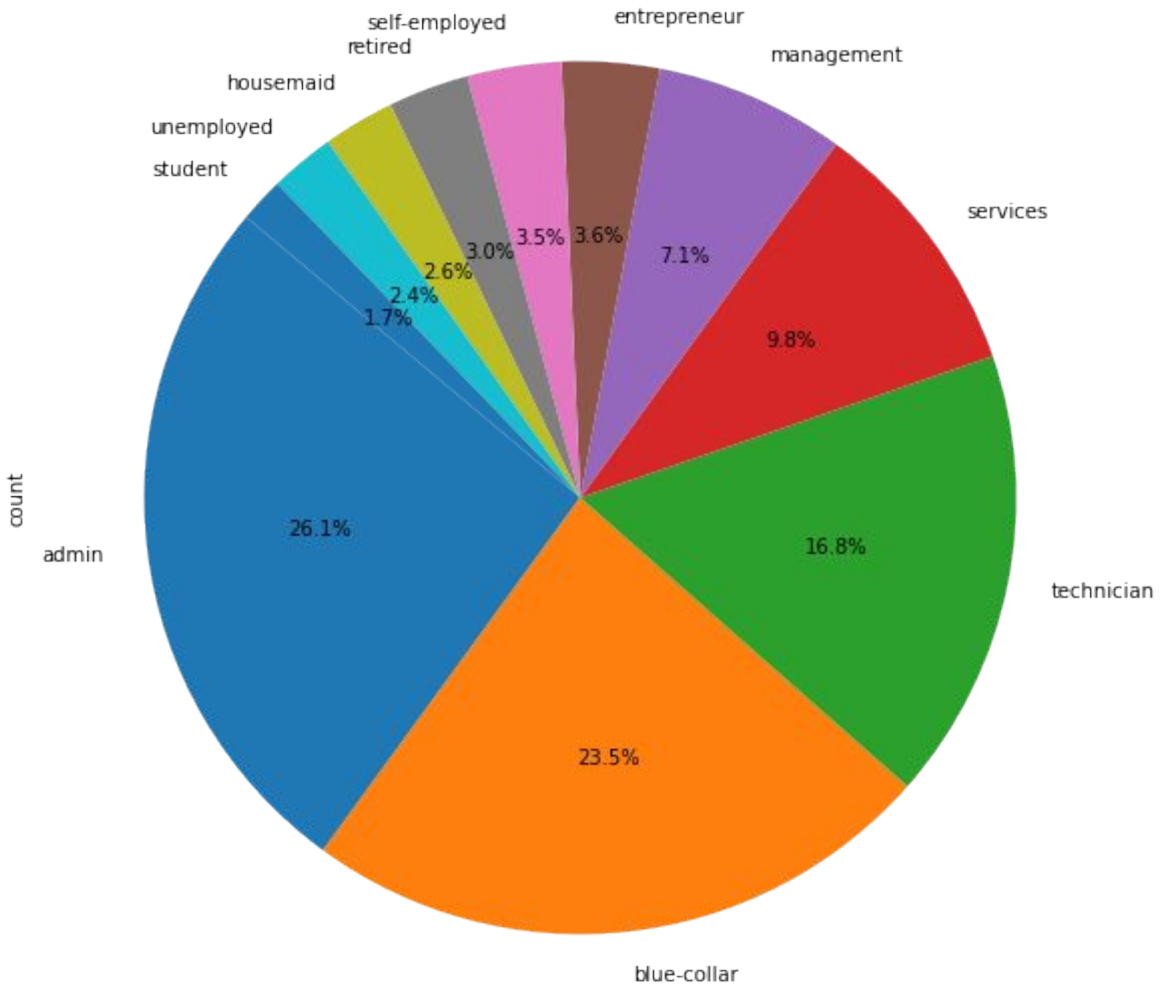


Education

Data Glacier
Your Deep Learning Partner

EDA

Distribution of Categorical Data

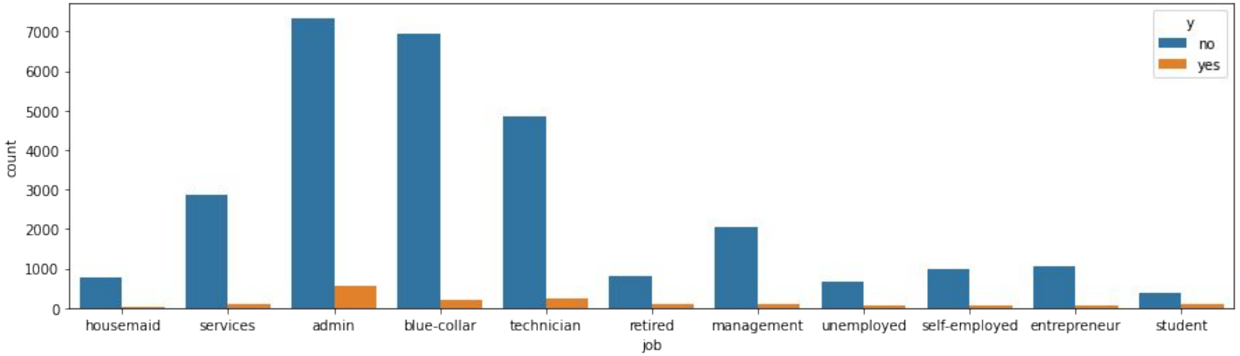


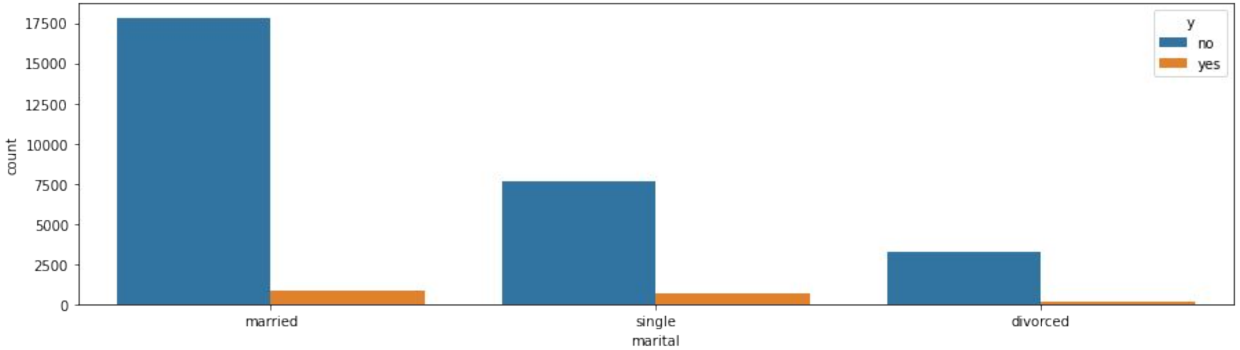
Employment

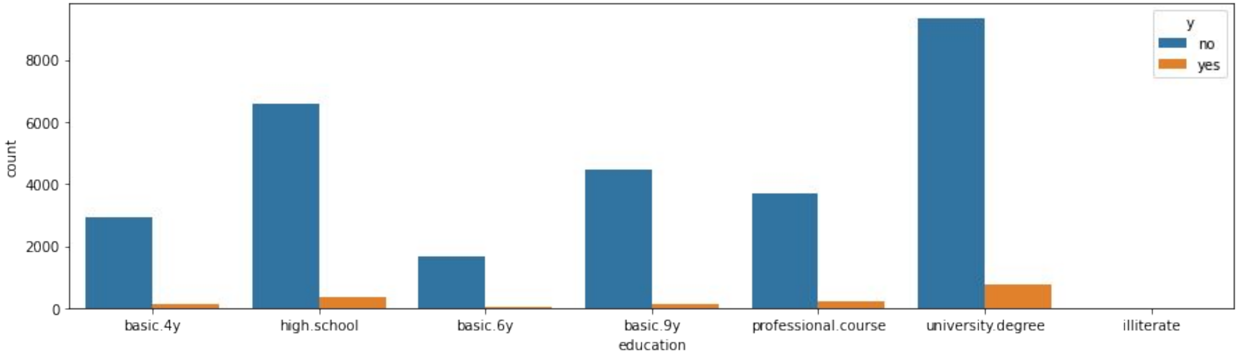
Data Glacier
Your Deep Learning Partner

Findings

Relationship of categorical data with output result





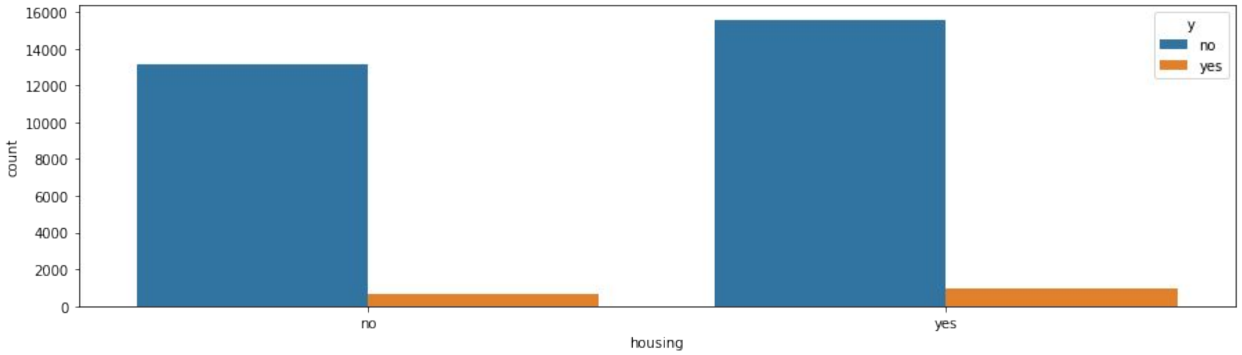


Data Glacier

Your Deep Learning Partner

Findings

Relationship of categorical data with output result

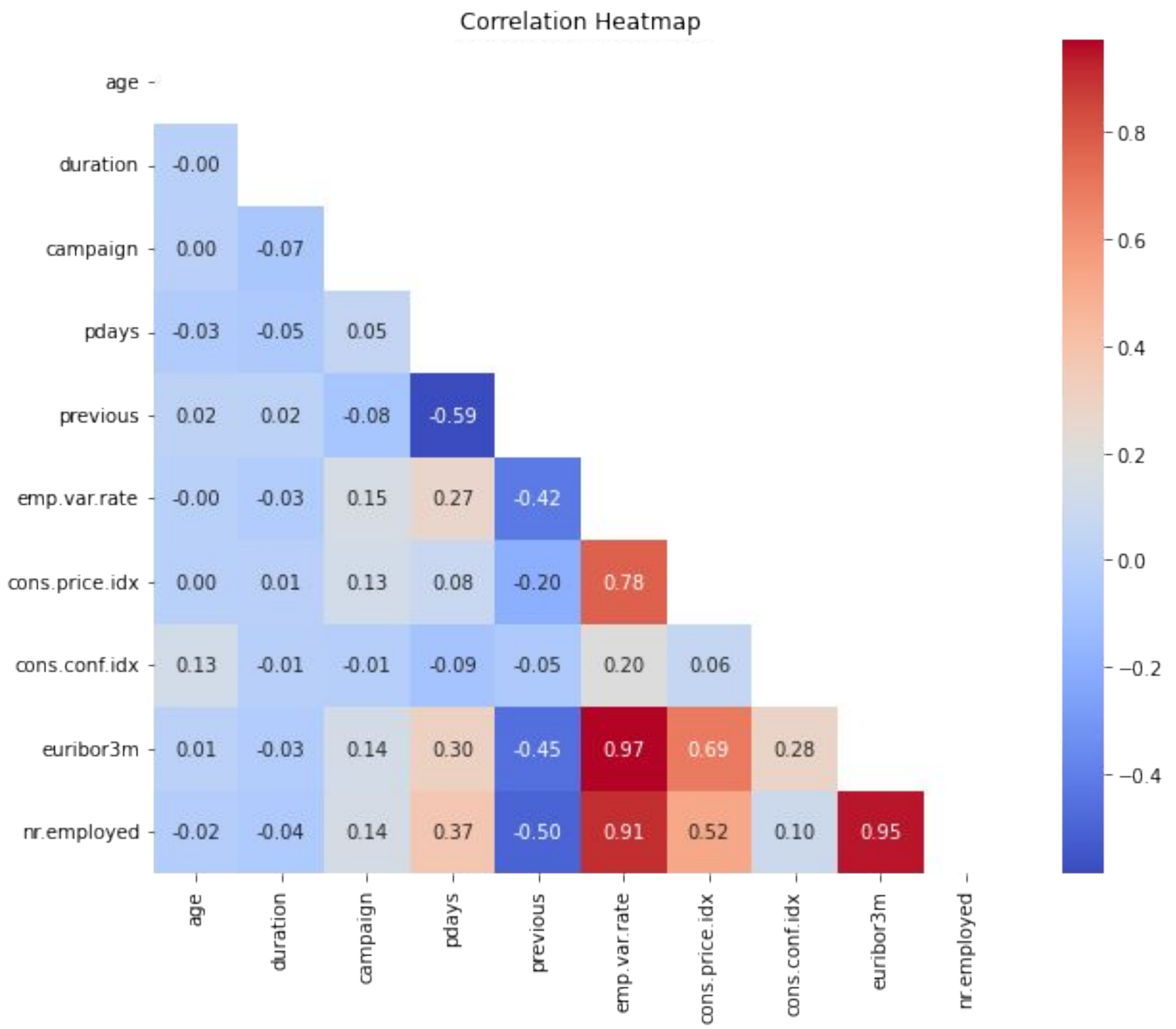


I plotted 4 bar charts containing the number of people subscribing to the terms with different ages, jobs, educations, loan, marital statuses, and I found out for each of these attributes, the number of people who do not subscribe to the terms surpasses the number of people who subscribe to it.

Data Glacier
Your Deep Learning Partner

Findings

Correlation Heatmap



Recommended Models

Recommended Models As our problem is to predict whether a customer will purchase the term deposit or not, an ideal solution would be to employ a binary classification model that excels in making accurate predictions.

Below are some models that we believe will be best suited for this problem. We will also outline how we will deal with imbalance and testing our models.

Logistic Regression: This model is straightforward, quick to train, and its output is highly interpretable. However, it assumes a linear relationship between features and the log-odds of the target variable, which might not always hold true.

Random Forest: It is an ensemble method that combines multiple decisions trees to improve accuracy and reduce overfitting,

Gradient Boosting Classifier: It sequentially builds trees to correct errors made by previous trees, making it effective for various table

AdaBoost Classifier: It combines weak learners into a strong learner by assigning higher weights to maintain data points, primarily used for binary classification.

Bagging Classifier: It creates multiple models from bootstraped data samples and averages their predictions to reduce variance and improve stability.

Extra Trees Classifier: Like Random Forest, uses decision trees but with random feature splits, making it computationally efficient.

Recommended Models

Support Vector Machines: Finds an optimal hyperplane to separate data points and can use different kernel functions for improved separation.

KNN: It classified data points based on the majority class among their k nearest neighbors, suitable for classification.

GaussianNB: It is a probabilistic classification model based on Bayes Theorem, assuming feature independence with GaussianNB using Gaussian Distributions.

Decision Tree Classifer: It makes decisions by recursively splitting data based on feature values, creating interpretable models for classification and regression tasks.

Handling Imbalance

Imbalance are a normal problem of binary classification models so we need to handles these are seemed fit.

Resampling: Adjust the class distribution by oversampling the minority class, undersampling the majority class, or using a combination of both. This helps create a more balanced dataset, but may lead to overfitting (oversampling) or loss of information (undersampling)

Evaluation of Model In order to assess the performance of our models, we will employ precision, recall, and F1-score as our evaluation metrics. Utilizing accuracy as a measure would not yield reliable results in the context of our current models, particularly when dealing with imbalanced dataset

Thank You!!!

www.thebodytransformation.com

Data Glacier

Your Deep Learning Partner