afuc6szj0

October 30, 2024

[2]:
```
import kagglehub

# Download latest version
path = kagglehub.dataset_download("janiobachmann/bank-marketing-dataset")

print("Path to dataset files:", path)
```

```
Downloading from
https://www.kaggle.com/api/v1/datasets/download/janiobachmann/bank-marketing-
dataset?dataset_version_number=1…

100%|      | 142k/142k [00:00<00:00, 403kB/s]

Extracting files…
Path to dataset files:
C:\Users\Spandana\.cache\kagglehub\datasets\janiobachmann\bank-marketing-
dataset\versions\1
```

[4]:
```
import pandas as pd
import numpy as np
from sklearn.preprocessing import OneHotEncoder, MinMaxScaler

filepath = 'bank.csv'

def load_data(filepath):
    """Load the dataset from a CSV file."""
    return pd.read_csv(filepath)

def handle_missing_values(df):
    """Handle missing values in the DataFrame."""
    # For simplicity, fill missing values with the mode for categorical columns
    for column in df.select_dtypes(include=['object']):
        df[column].fillna(df[column].mode()[0], inplace=True)

    # For numerical columns, fill with mean (customize as necessary)
    for column in df.select_dtypes(include=[np.number]):
        df[column].fillna(df[column].mean(), inplace=True)
```

```
    return df

def encode_categorical_variables(df):
    """Encode categorical variables using One-Hot Encoding."""
    categorical_cols = df.select_dtypes(include=['object']).columns
    df = pd.get_dummies(df, columns=categorical_cols, drop_first=True)
    return df

def scale_numerical_features(df):
    """Scale numerical features using Min-Max Scaling."""
    scaler = MinMaxScaler()
    numerical_cols = df.select_dtypes(include=[np.number]).columns
    df[numerical_cols] = scaler.fit_transform(df[numerical_cols])
    return df

def preprocess_data(filepath):
    """Load, clean, and preprocess the data."""
    df = load_data(filepath)
    df = handle_missing_values(df)
    df = encode_categorical_variables(df)
    df = scale_numerical_features(df)
    return df

df_preprocessed = preprocess_data(filepath)
print(df_preprocessed.head())
```

```
        age   balance       day  duration  campaign  pdays  previous  \
0  0.532468  0.104371  0.133333  0.268110  0.000000    0.0       0.0
1  0.493506  0.078273  0.133333  0.377675  0.000000    0.0       0.0
2  0.298701  0.092185  0.133333  0.357566  0.000000    0.0       0.0
3  0.480519  0.105882  0.133333  0.148750  0.000000    0.0       0.0
4  0.467532  0.079851  0.133333  0.172983  0.016129    0.0       0.0

   job_blue-collar  job_entrepreneur  job_housemaid  …  month_jun  \
0            False             False          False  …      False
1            False             False          False  …      False
2            False             False          False  …      False
3            False             False          False  …      False
4            False             False          False  …      False

   month_mar  month_may  month_nov  month_oct  month_sep  poutcome_other  \
0      False       True      False      False      False           False
1      False       True      False      False      False           False
2      False       True      False      False      False           False
3      False       True      False      False      False           False
4      False       True      False      False      False           False
```

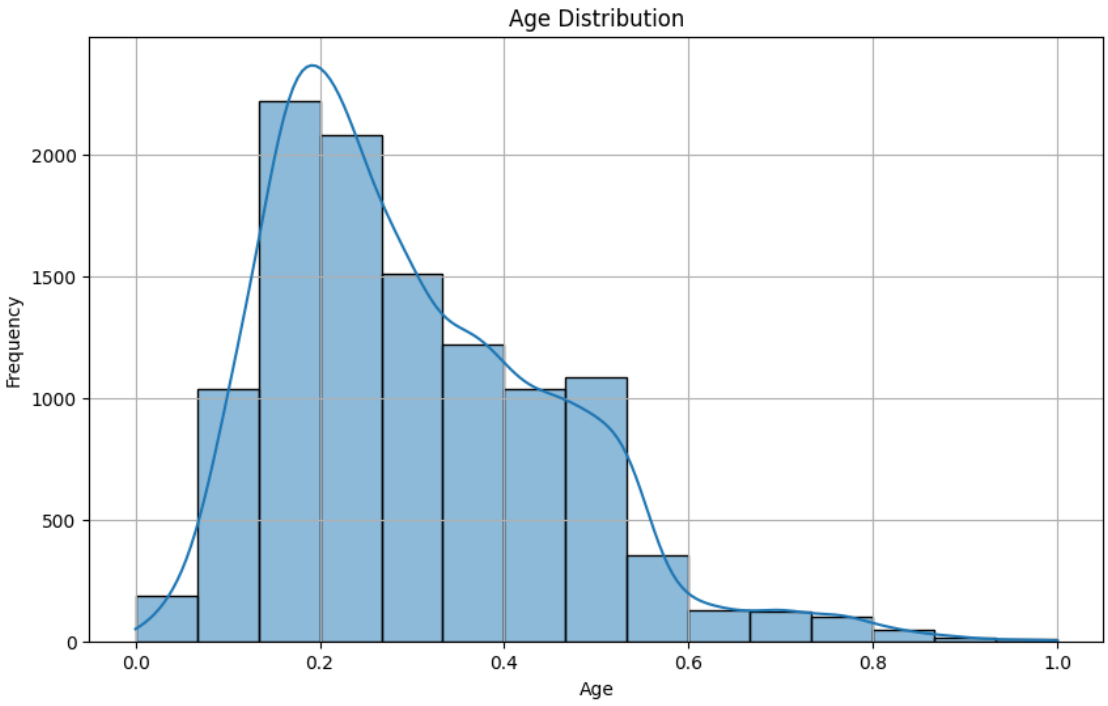
```
   poutcome_success  poutcome_unknown  deposit_yes
0             False              True         True
1             False              True         True
2             False              True         True
3             False              True         True
4             False              True         True

[5 rows x 43 columns]
```

```
C:\Users\Spandana\AppData\Local\Temp\ipykernel_10568\3193765478.py:15:
FutureWarning: A value is trying to be set on a copy of a DataFrame or Series
through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  df[column].fillna(df[column].mode()[0], inplace=True)
C:\Users\Spandana\AppData\Local\Temp\ipykernel_10568\3193765478.py:19:
FutureWarning: A value is trying to be set on a copy of a DataFrame or Series
through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  df[column].fillna(df[column].mean(), inplace=True)
```

[5]:
```python
import matplotlib.pyplot as plt
import seaborn as sns

def plot_age_distribution(df):
    """Plot the distribution of ages in the dataset."""
    plt.figure(figsize=(10, 6))
    sns.histplot(df['age'], bins=15, kde=True)
    plt.title('Age Distribution')
    plt.xlabel('Age')
    plt.ylabel('Frequency')
    plt.grid()
```

```
    plt.show()

def plot_job_distribution(df):
    """Plot the distribution of job types in the dataset."""
    plt.figure(figsize=(12, 6))
    sns.countplot(y='job', data=df, order=df['job'].value_counts().index)
    plt.title('Job Type Distribution')
    plt.xlabel('Count')
    plt.ylabel('Job Type')
    plt.grid()
    plt.show()

def plot_marital_status_distribution(df):
    """Plot the distribution of marital status in the dataset."""
    plt.figure(figsize=(8, 6))
    sns.countplot(x='marital', data=df)
    plt.title('Marital Status Distribution')
    plt.xlabel('Marital Status')
    plt.ylabel('Count')
    plt.grid()
    plt.show()

def plot_education_distribution(df):
    """Plot the distribution of education levels in the dataset."""
    plt.figure(figsize=(10, 6))
    sns.countplot(x='education', data=df, order=df['education'].value_counts().
 ↪index)
    plt.title('Education Level Distribution')
    plt.xlabel('Education Level')
    plt.ylabel('Count')
    plt.grid()
    plt.show()

def plot_balance_boxplot(df):
    """Plot a box plot of account balance grouped by housing loan status."""
    plt.figure(figsize=(10, 6))
    sns.boxplot(x='housing', y='balance', data=df)
    plt.title('Account Balance by Housing Loan Status')
    plt.xlabel('Housing Loan (Yes/No)')
    plt.ylabel('Balance')
    plt.grid()
    plt.show()

def plot_correlation_heatmap(df):
    """Plot a heatmap of the correlation matrix for numerical features."""
    plt.figure(figsize=(12, 8))
    correlation_matrix = df.corr()
```

```
    sns.heatmap(correlation_matrix, annot=True, fmt='.2f', cmap='coolwarm',␣
 ↪square=True)
    plt.title('Correlation Heatmap')
    plt.show()

def plot_contact_duration_scatter(df):
    """Plot a scatter plot of contact duration against the number of contacts␣
 ↪in the campaign."""
    plt.figure(figsize=(10, 6))
    sns.scatterplot(x='duration', y='campaign', data=df)
    plt.title('Contact Duration vs. Number of Contacts in Campaign')
    plt.xlabel('Contact Duration (seconds)')
    plt.ylabel('Number of Contacts in Campaign')
    plt.grid()
    plt.show()


df = df_preprocessed.copy()

plot_age_distribution(df)
plot_job_distribution(df)
plot_marital_status_distribution(df)
plot_education_distribution(df)
plot_balance_boxplot(df)
plot_correlation_heatmap(df)
plot_contact_duration_scatter(df)
```



```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
File c:\Users\Spandana\miniconda3\Lib\site-packages\pandas\core\indexes\base.py
 ↪3802, in Index.get_loc(self, key)
   3801 try:
-> 3802     return self._engine.get_loc(casted_key)
   3803 except KeyError as err:

File index.pyx:153, in pandas._libs.index.IndexEngine.get_loc()

File index.pyx:182, in pandas._libs.index.IndexEngine.get_loc()

File pandas\\_libs\\hashtable_class_helper.pxi:7081, in pandas._libs.hashtable.
 ↪PyObjectHashTable.get_item()

File pandas\\_libs\\hashtable_class_helper.pxi:7089, in pandas._libs.hashtable.
 ↪PyObjectHashTable.get_item()

KeyError: 'job'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
Cell In[5], line 76
     73 df = df_preprocessed.copy()
     75 plot_age_distribution(df)
---> 76 plot_job_distribution(df)
     77 plot_marital_status_distribution(df)
     78 plot_education_distribution(df)

Cell In[5], line 17, in plot_job_distribution(df)
     15 """Plot the distribution of job types in the dataset."""
     16 plt.figure(figsize=(12, 6))
---> 17 sns.countplot(y='job', data=df, order=df['job'].value_counts().index)
     18 plt.title('Job Type Distribution')
     19 plt.xlabel('Count')

File c:\Users\Spandana\miniconda3\Lib\site-packages\pandas\core\frame.py:4090,␣
 ↪in DataFrame.__getitem__(self, key)
   4088 if self.columns.nlevels > 1:
   4089     return self._getitem_multilevel(key)
-> 4090 indexer = self.columns.get_loc(key)
   4091 if is_integer(indexer):
   4092     indexer = [indexer]
```

```
File c:\Users\Spandana\miniconda3\Lib\site-packages\pandas\core\indexes\base.py
 ↪3809, in Index.get_loc(self, key)
   3804     if isinstance(casted_key, slice) or (
   3805         isinstance(casted_key, abc.Iterable)
   3806         and any(isinstance(x, slice) for x in casted_key)
   3807     ):
   3808         raise InvalidIndexError(key)
-> 3809     raise KeyError(key) from err
   3810 except TypeError:
   3811     # If we have a listlike key, _check_indexing_error will raise
   3812     #  InvalidIndexError. Otherwise we fall through and re-raise
   3813     #  the TypeError.
   3814     self._check_indexing_error(key)

KeyError: 'job'
```

```
<Figure size 1200x600 with 0 Axes>
```

[ ]: