EDA

August 11, 2025

#

EDA

[20]:
```
# Import libraries
import pandas as pd
import numpy as np
import os
import matplotlib.pyplot as plt
import matplotlib
import seaborn as sns
import tarfile
from sklearn.datasets import make_classification
from sklearn.preprocessing import (
    OneHotEncoder,
    OrdinalEncoder,
    LabelEncoder,
    StandardScaler,
    Normalizer,
)

# from sklearn.metrics import confusion_matrix,␣
 ↪accuracy_score,plot_confusion_matrix, classification_report
from sklearn.metrics import (
    confusion_matrix,
    accuracy_score,
    ConfusionMatrixDisplay,
    classification_report,
)
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_selection import SelectKBest, chi2
from sklearn.linear_model import SGDClassifier
```

[21]:
```
# Retrieve dataset
# Code developed with assistance from Generative AI

# check development environment
```

```python
try:
    import google.colab
    IN_COLAB = True
except:
    IN_COLAB = False

# connect to drive if IN_COLAB = True
if IN_COLAB:
    from google.colab import drive
    drive.mount('/content/drive')

# set datapath
data_path = "../datasets/bank-additional-full.csv"

# download dataset if IN_COLAB = True
if IN_COLAB:
    print("Running in Colab... downloading dataset.")
    data_url = "https://raw.githubusercontent.com/junclemente/
↪ads504-final_project/main/datasets/bank-additional-full.csv"
    if not os.path.exists("../datasets"):
        os.makedirs("../datasets")
    if not os.path.exists(data_path):
        !wget -q {data_url} -O {data_path}
else:
    print("Running locally... using local dataset.")

# Load the dataset
df = pd.read_csv(data_path, sep=";")
```

```
Running locally… using local dataset.
```

[22]:
```python
df.head()
```

[22]:
```
   age        job  marital    education  default housing loan    contact  \
0   56  housemaid  married     basic.4y       no      no   no  telephone
1   57   services  married  high.school  unknown      no   no  telephone
2   37   services  married  high.school       no     yes   no  telephone
3   40     admin.  married     basic.6y       no      no   no  telephone
4   56   services  married  high.school       no      no  yes  telephone

  month day_of_week  …  campaign  pdays  previous     poutcome emp.var.rate  \
0   may         mon  …         1    999         0  nonexistent          1.1
1   may         mon  …         1    999         0  nonexistent          1.1
2   may         mon  …         1    999         0  nonexistent          1.1
3   may         mon  …         1    999         0  nonexistent          1.1
4   may         mon  …         1    999         0  nonexistent          1.1
```

```
   cons.price.idx  cons.conf.idx  euribor3m  nr.employed   y
0          93.994          -36.4      4.857       5191.0  no
1          93.994          -36.4      4.857       5191.0  no
2          93.994          -36.4      4.857       5191.0  no
3          93.994          -36.4      4.857       5191.0  no
4          93.994          -36.4      4.857       5191.0  no

[5 rows x 21 columns]
```

0.1 1. Basic Information

[23]:
```python
# Basic information
print("Shape of dataset:", df.shape)
print("\nData types:\n", df.dtypes)
```

```
Shape of dataset: (41188, 21)

Data types:
 age                 int64
job                object
marital            object
education          object
default            object
housing            object
loan               object
contact            object
month              object
day_of_week        object
duration            int64
campaign            int64
pdays               int64
previous            int64
poutcome           object
emp.var.rate      float64
cons.price.idx    float64
cons.conf.idx     float64
euribor3m         float64
nr.employed       float64
y                  object
dtype: object
```

0.2 2. Missing Values and Unique Summary

[24]:
```python
# Missing values
print(df.isnull().sum())
```

```
age               0
job               0
```

```
marital           0
education         0
default           0
housing           0
loan              0
contact           0
month             0
day_of_week       0
duration          0
campaign          0
pdays             0
previous          0
poutcome          0
emp.var.rate      0
cons.price.idx    0
cons.conf.idx     0
euribor3m         0
nr.employed       0
y                 0
dtype: int64
```

[25]:
```python
# Any duplicates?
print(df.nunique())
```

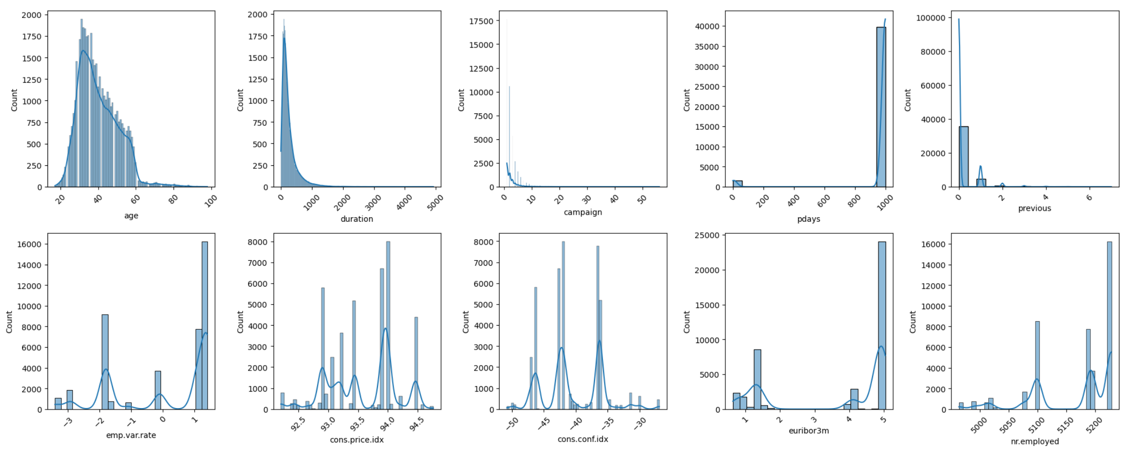
```
age                 78
job                 12
marital              4
education            8
default              3
housing              3
loan                 3
contact              2
month               10
day_of_week          5
duration          1544
campaign            42
pdays               27
previous             8
poutcome             3
emp.var.rate        10
cons.price.idx      26
cons.conf.idx       26
euribor3m          316
nr.employed         11
y                    2
dtype: int64
```

0.3 3. Univariate Analysis (Numerical Features)

[26]:
```
df.describe()
```

[26]:
```
                age      duration      campaign         pdays      previous  \
count  41188.00000  41188.000000  41188.000000  41188.000000  41188.000000
mean      40.02406    258.285010      2.567593    962.475454      0.172963
std       10.42125    259.279249      2.770014    186.910907      0.494901
min       17.00000      0.000000      1.000000      0.000000      0.000000
25%       32.00000    102.000000      1.000000    999.000000      0.000000
50%       38.00000    180.000000      2.000000    999.000000      0.000000
75%       47.00000    319.000000      3.000000    999.000000      0.000000
max       98.00000   4918.000000     56.000000    999.000000      7.000000

       emp.var.rate  cons.price.idx  cons.conf.idx     euribor3m   nr.employed
count  41188.000000    41188.000000   41188.000000  41188.000000  41188.000000
mean       0.081886       93.575664     -40.502600      3.621291   5167.035911
std        1.570960        0.578840       4.628198      1.734447     72.251528
min       -3.400000       92.201000     -50.800000      0.634000   4963.600000
25%       -1.800000       93.075000     -42.700000      1.344000   5099.100000
50%        1.100000       93.749000     -41.800000      4.857000   5191.000000
75%        1.400000       93.994000     -36.400000      4.961000   5228.100000
max        1.400000       94.767000     -26.900000      5.045000   5228.100000
```

[27]:
```
# Boxplot and histogram for a few important features
num_cols = df.select_dtypes(include="number").columns
n_cols = 5
n_rows = 2

fig, axes = plt.subplots(n_rows, n_cols, figsize=(20, 8))
axes = axes.flatten()

for i, col in enumerate(num_cols[: n_cols * n_rows]):
    sns.histplot(df[col], kde=True, ax=axes[i])
    axes[i].tick_params(axis="x", rotation=45)
plt.tight_layout()
plt.show()
```



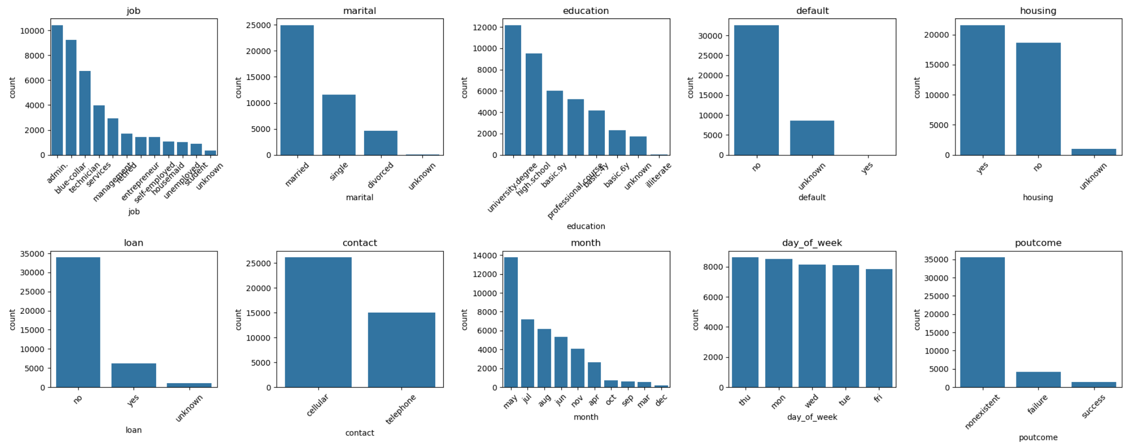
0.4 4. Univariate Analysis (Categorical Features)

[28]:
```python
# Summary statistics (categorical features)
print("\nSummary statistics (categorical):\n", df.describe(include="object"))
```

```
Summary statistics (categorical):
            job  marital          education default housing   loan   contact  \
count    41188    41188              41188   41188   41188  41188     41188
unique      12        4                  8       3       3      3         2
top     admin.  married  university.degree      no     yes     no  cellular
freq     10422    24928              12168   32588   21576  33950     26144

        month day_of_week     poutcome      y
count   41188       41188        41188  41188
unique     10           5            3      2
top       may         thu  nonexistent     no
freq    13769        8623        35563  36548
```

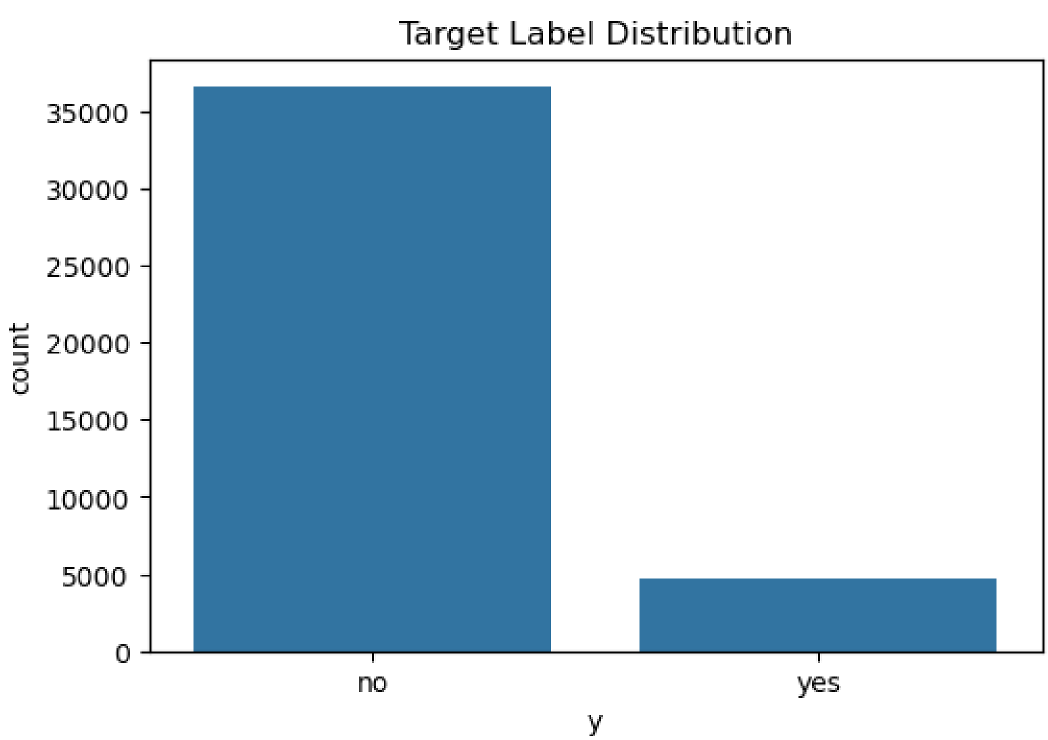
[29]:
```python
cat_cols = df.select_dtypes(include="object").columns.drop("y")
n_cols = 5
n_rows = 2

fig, axes = plt.subplots(n_rows, n_cols, figsize=(20, 8))
axes = axes.flatten()

for i, col in enumerate(cat_cols):
    sns.countplot(x=col, data=df, order=df[col].value_counts().index,␣
 ↪ax=axes[i])
    axes[i].set_title(f"{col}", fontsize=12)
    axes[i].tick_params(axis="x", rotation=45)
```

```
plt.tight_layout()
plt.show()
```



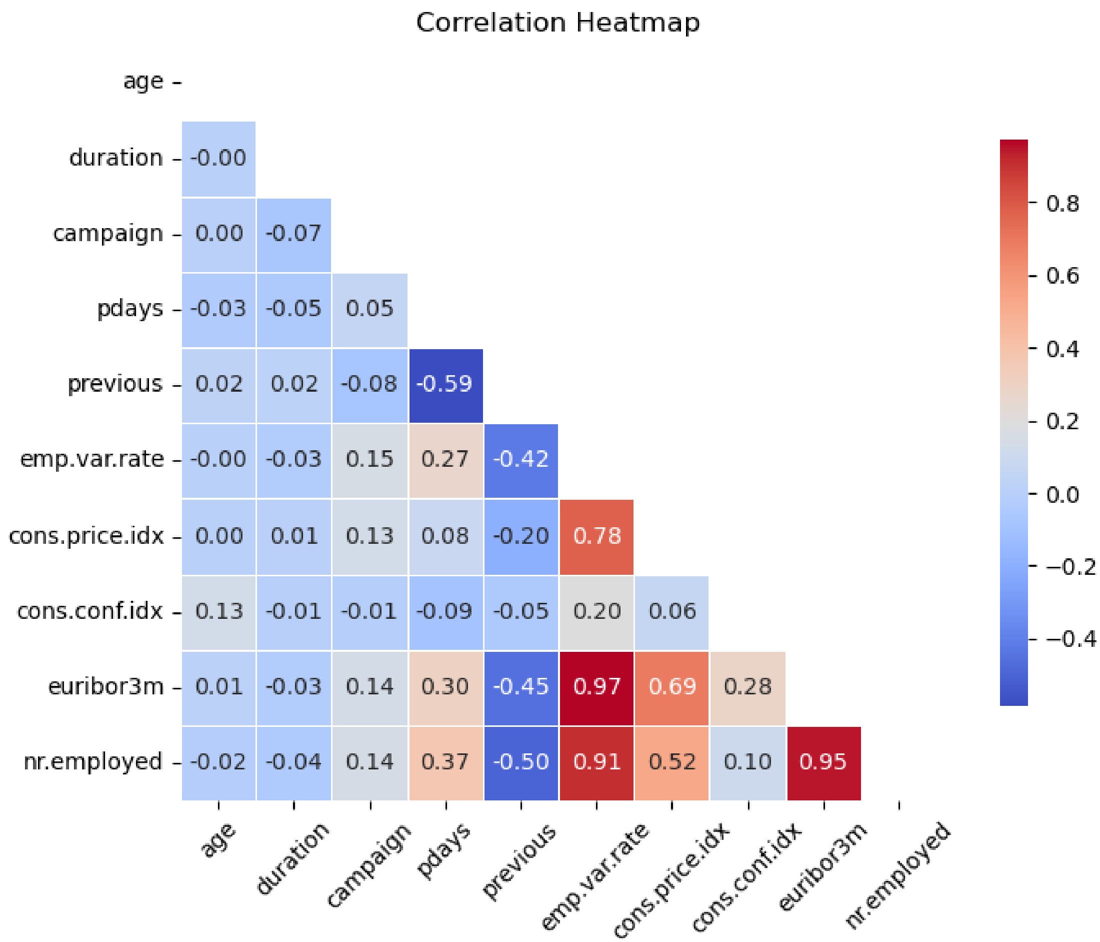
0.5 5. Target Variable (y) Distribution

[30]:
```
# Plot target variable distribution
plt.figure(figsize=(6, 4))
sns.countplot(data=df, x="y")
plt.title("Target Label Distribution")
plt.show()
```



0.6 6. Correlation heatmap (only for numeric columns)

[31]:
```
# Set matrix
corr = df.corr(numeric_only=True)
# Create a mask for the upper triangle plot
mask = np.triu(np.ones_like(corr, dtype=bool))

# set up the matplotlib figure
plt.figure(figsize=(10, 6))

sns.heatmap(
    corr,
    mask=mask,
    cmap="coolwarm",
    annot=True,
    fmt=".2f",
    square=True,
    linewidths=0.5,
    cbar_kws={"shrink": 0.75},
)
plt.xticks(rotation=45)
plt.title("Correlation Heatmap")
plt.show()
```



0.7 7. Correlatioin with Target label

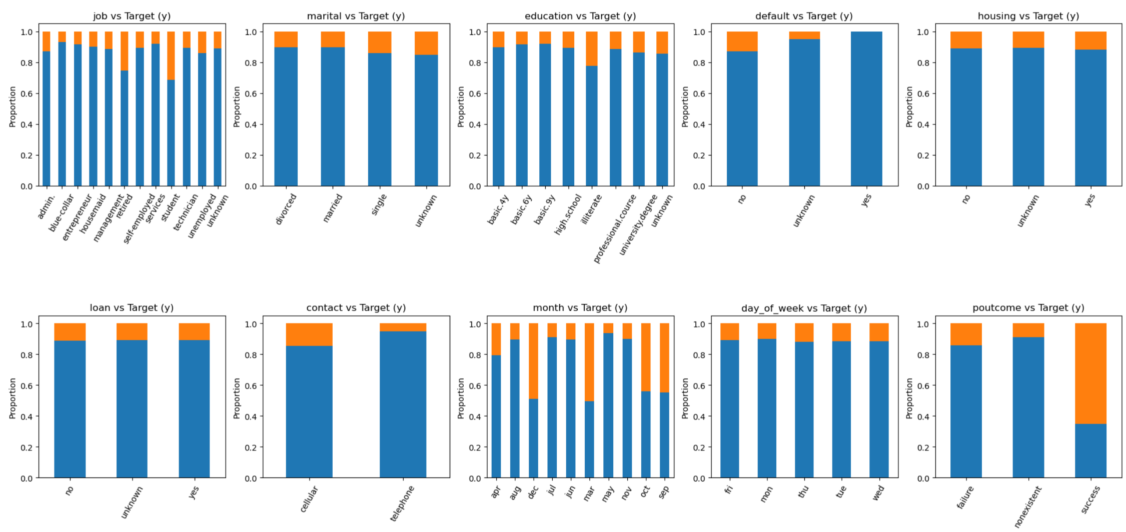
[32]:
```
# Encode target to numeric for correlation
df_temp = df.copy()
df_temp["y"] = df_temp["y"].map({"no": 0, "yes": 1})
print(df_temp.corr(numeric_only=True)["y"].sort_values(ascending=False))
```

```
y                 1.000000
duration          0.405274
previous          0.230181
cons.conf.idx     0.054878
age               0.030399
campaign         -0.066357
cons.price.idx   -0.136211
emp.var.rate     -0.298334
euribor3m        -0.307771
pdays            -0.324914
nr.employed      -0.354678
```

Name: y, dtype: float64

0.8 8. Bivariate Analysis: Categorical vs Target

[33]:
```
cat_cols = df.select_dtypes(include="object").columns.drop("y")
n_cols = 5
n_rows = 2

fig, axes = plt.subplots(n_rows, n_cols, figsize=(24, 10))
axes = axes.flatten()


for i, col in enumerate(cat_cols[: n_cols * n_rows]):
    ct = pd.crosstab(df[col], df["y"], normalize="index")
    ct.plot(kind="bar", stacked=True, ax=axes[i], legend=False)
    axes[i].set_title(f"{col} vs Target (y)", fontsize=12)
    axes[i].set_ylabel("Proportion")
    axes[i].tick_params(axis="x", rotation=60)
    axes[i].set_xlabel("")

fig.subplots_adjust(hspace=0.8, bottom=0.1)

plt.show()
```



0.8.1 Chi2 Contingency Table

[34]:
```
from scipy.stats import chi2_contingency
from itertools import combinations
```

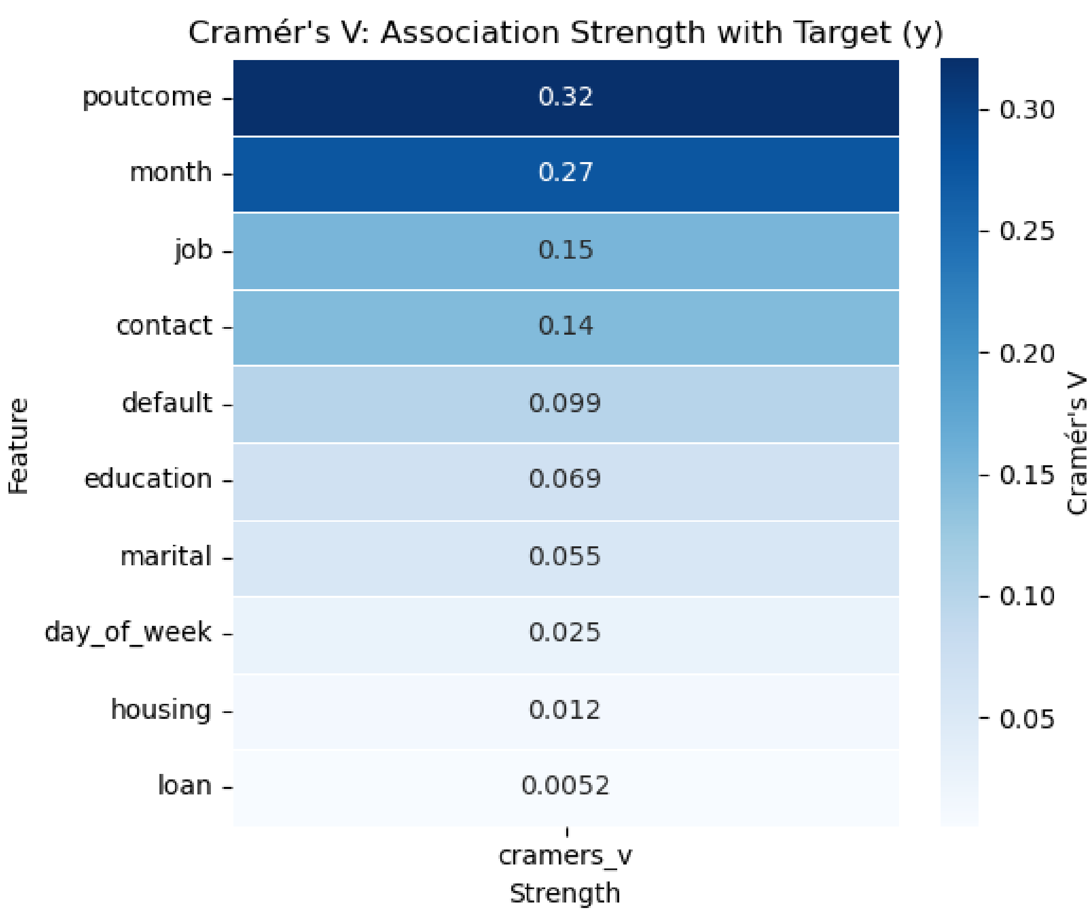
```python
# get categorical columns
cat_cols = df.select_dtypes(include=["object", "category"]).columns.tolist()
# display(cat_cols)

# drop target column 'y'
cat_cols.remove("y")


# Cramer's V function
def cramer_v_score(x, y):
    table = pd.crosstab(x, y)
    chi2 = chi2_contingency(table)[0]
    n = table.sum().sum()
    return np.sqrt(chi2 / (n * (min(table.shape) - 1)))


results = []

for col in cat_cols:
    table = pd.crosstab(df[col], df["y"])

    # run chi2 test
    chi2, p, dof, expected = chi2_contingency(table)

    # cramer's v score
    cramers_score = cramer_v_score(df[col], df["y"])

    results.append(
        {
            "feature": col,
            "chi2_stat": chi2.round(4),
            "dof": dof,
            "p_value": p.round(4),
            "significant (p-value)": p < 0.05,
            "cramers_v": cramers_score.round(4),
        }
    )

results_df = pd.DataFrame(results)
results_df.sort_values(by="cramers_v", ascending=False, inplace=True)
display(results_df)
```

	feature	chi2_stat	dof	p_value	significant (p-value)	cramers_v
9	poutcome	4230.5238	2	0.0000	True	0.3205
7	month	3101.1494	9	0.0000	True	0.2744
0	job	961.2424	11	0.0000	True	0.1528
6	contact	862.3184	1	0.0000	True	0.1447
3	default	406.5775	2	0.0000	True	0.0994

2	education	193.1059	7	0.0000	True	0.0685
1	marital	122.6552	3	0.0000	True	0.0546
8	day_of_week	26.1449	4	0.0000	True	0.0252
4	housing	5.6845	2	0.0583	False	0.0117
5	loan	1.0940	2	0.5787	False	0.0052

Features with p-value < 0.05 are significantly associated with the target variable (y). These should be considered for further analysis or modeling.

[35]:
```python
plt.figure(figsize=(6, len(results_df) * 0.5))
sns.heatmap(
    results_df[["cramers_v"]].set_index(results_df["feature"]),
    annot=True,
    cmap="Blues",
    linewidths=0.5,
    cbar_kws={"label": "Cramér's V"},
)
plt.title("Cramér's V: Association Strength with Target (y)")
plt.xlabel("Strength")
plt.ylabel("Feature")
plt.tight_layout()
plt.show()
```



Features with Cramers V > 0.1 are significantly associated with the target variable (y). These should be considered for further analysis or modeling.