1_exploratory_data_analysis

November 2, 2020

1 Exploratory Data Analysis

[1]:
```
# Install watermark to print system and hardware infomation
# %conda install watermark
%load_ext watermark
%watermark -v -m -p numpy,pandas,matplotlib,seaborn
```

```
CPython 3.8.6
IPython 7.18.1

numpy 1.19.2
pandas 1.1.3
matplotlib 3.3.2
seaborn 0.11.0

compiler   : Clang 10.0.1
system     : Darwin
release    : 19.6.0
machine    : x86_64
processor  : i386
CPU cores  : 16
interpreter: 64bit
```

1.1 Import Data

The first step of data preparation is to import data. We use `pandas`'s `read_csv()` to import data and take care of data types, true/false values and missing values.

[1]:
```
import numpy as np
import pandas as pd
```

[2]:
```
def import_dataset(filename):
    bank_mkt = pd.read_csv(filename,
                           na_values=["unknown", "nonexistent"],
                           true_values=["yes", "success"],
                           false_values=["no", "failure"])
    # Treat pdays = 999 as missing values
    bank_mkt["pdays"] = bank_mkt["pdays"].replace(999, pd.NA)
```

```
    # Convert types, "Int64" is nullable integer data type in pandas
    bank_mkt = bank_mkt.astype(dtype={"age": "Int64",
                                      "job": "category",
                                      "marital": "category",
                                      "education": "category",
                                      "default": "boolean",
                                      "housing": "boolean",
                                      "loan": "boolean",
                                      "contact": "category",
                                      "month": "category",
                                      "day_of_week": "category",
                                      "duration": "Int64",
                                      "campaign": "Int64",
                                      "pdays": "Int64",
                                      "previous": "Int64",
                                      "poutcome": "boolean",
                                      "y": "boolean"})
    # reorder categorical data
    bank_mkt["education"] = bank_mkt["education"].cat.
↪reorder_categories(["illiterate", "basic.4y", "basic.6y", "basic.9y", "high.
↪school", "professional.course", "university.degree"], ordered=True)
    bank_mkt["month"] = bank_mkt["month"].cat.reorder_categories(["mar", "apr",␣
↪"jun", "jul", "may", "aug", "sep", "oct", "nov", "dec"], ordered=True)
    bank_mkt["day_of_week"] = bank_mkt["day_of_week"].cat.
↪reorder_categories(["mon", "tue", "wed", "thu", "fri"], ordered=True)
    return bank_mkt
```

[3]:
```
bank_mkt = import_dataset("../data/BankMarketing.csv")
```

1.2 Exploratory Data Analysis

Exploratory Data Analysis is a process to explore the dataset with no assumptions or hypothesis. The objective is to give us enough insights for the future work.

There are many visualization libraries in Python. Pandas has its own plot API based on matplotlib and we will also use Seaborn and Altair. Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics. Altair is a declarative statistical visualization library for Python, based on Vega and Vega-Lite. Both libraries provide easy to use APIs and produce beautiful graphs.

[4]:
```
import altair as alt
import matplotlib.pyplot as plt
# cosmetic options for matplotlib
plt.style.use("seaborn")
plt.rcParams["figure.figsize"] = (6.4, 4.8)
plt.rcParams["figure.dpi"] = 300
plt.rcParams["axes.titleweight"] = "bold"
```

```
plt.rcParams["axes.titlepad"] = 10.0
plt.rcParams["axes.titlelocation"] = "left"
from IPython.display import set_matplotlib_formats
set_matplotlib_formats("svg")
import seaborn as sns
```

Let's first inpect the outcome distribution. As we can see below, the dataset is imbalanced. With 41188 rows of data, only 11.2% have positive outcome.

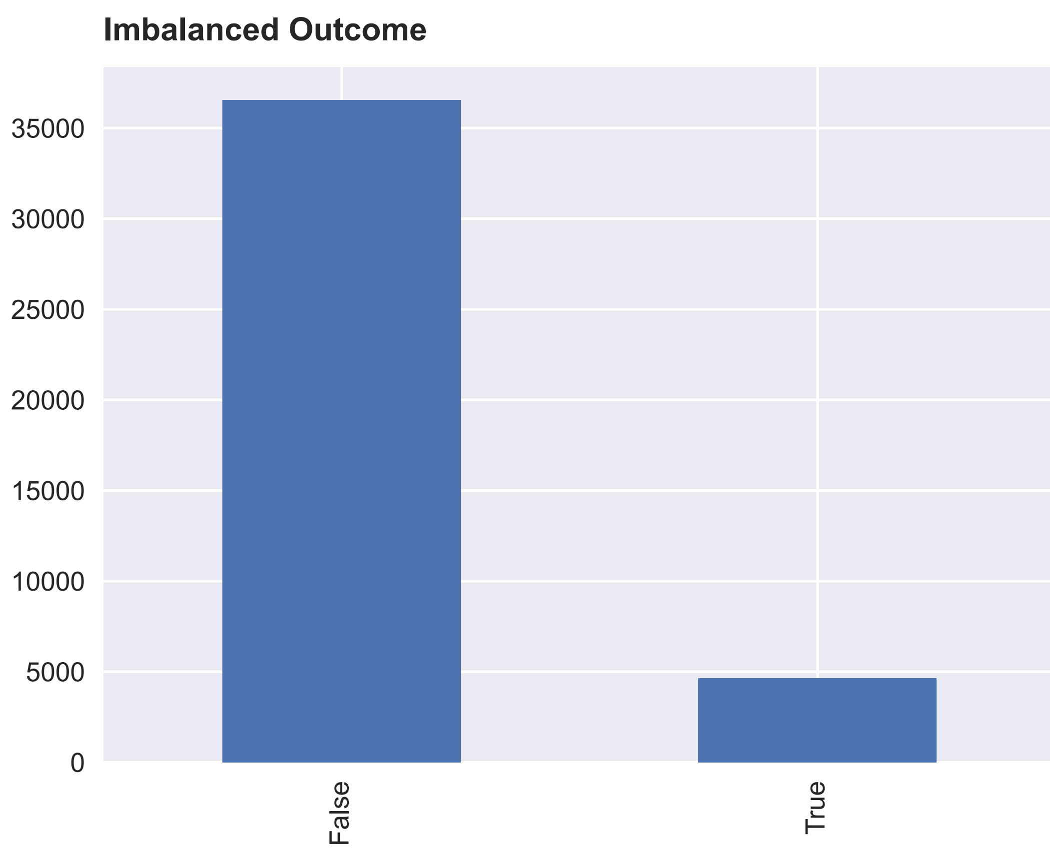
[5]:
```
bank_mkt["y"].count()
```

[5]: 41188

[6]:
```
bank_mkt["y"].sum()/bank_mkt["y"].count()
```

[6]: 0.11265417111780131

[7]:
```
y_count = bank_mkt["y"].value_counts().plot(kind = "bar", title="Imbalanced␣
↪Outcome")
```



Using `info()` we can see that most of features concerning the client are categorical/boolean type. And some fields such as `job`, `marital`, `education`, etc. are missing.

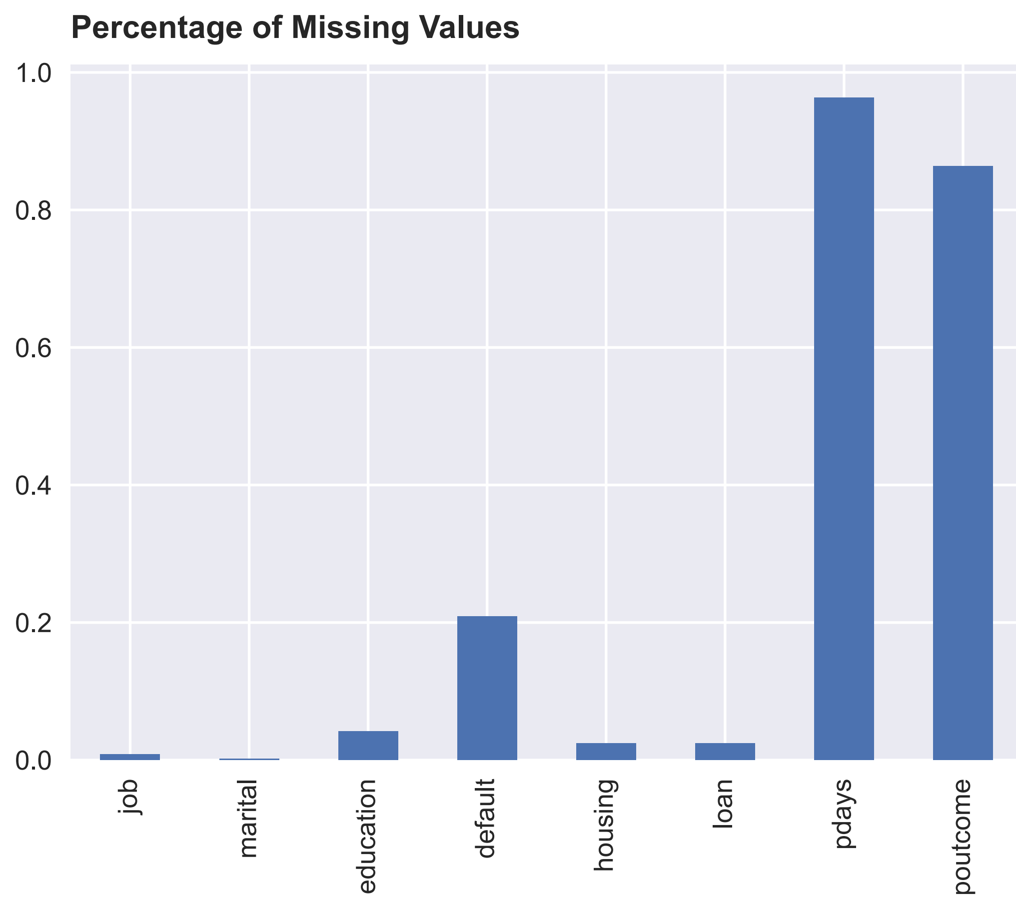
[8]:
```
bank_mkt.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41188 entries, 0 to 41187
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             41188 non-null  Int64
 1   job             40858 non-null  category
 2   marital         41108 non-null  category
 3   education       39457 non-null  category
 4   default         32591 non-null  boolean
 5   housing         40198 non-null  boolean
 6   loan            40198 non-null  boolean
 7   contact         41188 non-null  category
 8   month           41188 non-null  category
 9   day_of_week     41188 non-null  category
 10  duration        41188 non-null  Int64
 11  campaign        41188 non-null  Int64
 12  pdays           1515 non-null   Int64
 13  previous        41188 non-null  Int64
 14  poutcome        5625 non-null   boolean
 15  emp.var.rate    41188 non-null  float64
 16  cons.price.idx  41188 non-null  float64
 17  cons.conf.idx   41188 non-null  float64
 18  euribor3m       41188 non-null  float64
 19  nr.employed     41188 non-null  float64
 20  y               41188 non-null  boolean
dtypes: Int64(5), boolean(5), category(6), float64(5)
memory usage: 4.0 MB
```

1.2.1 Missing values

By checking the number of missing values, we can see nearly all client do not have `pdays` and `poutcome`. 20% of the clients do not have information of `default`.

[9]:
```
na = bank_mkt.isna().sum()
na_nonzero = na[na != 0]
na_perc = na_nonzero/bank_mkt.y.count()
na_bar = na_perc.plot.bar(title="Percentage of Missing Values")
```

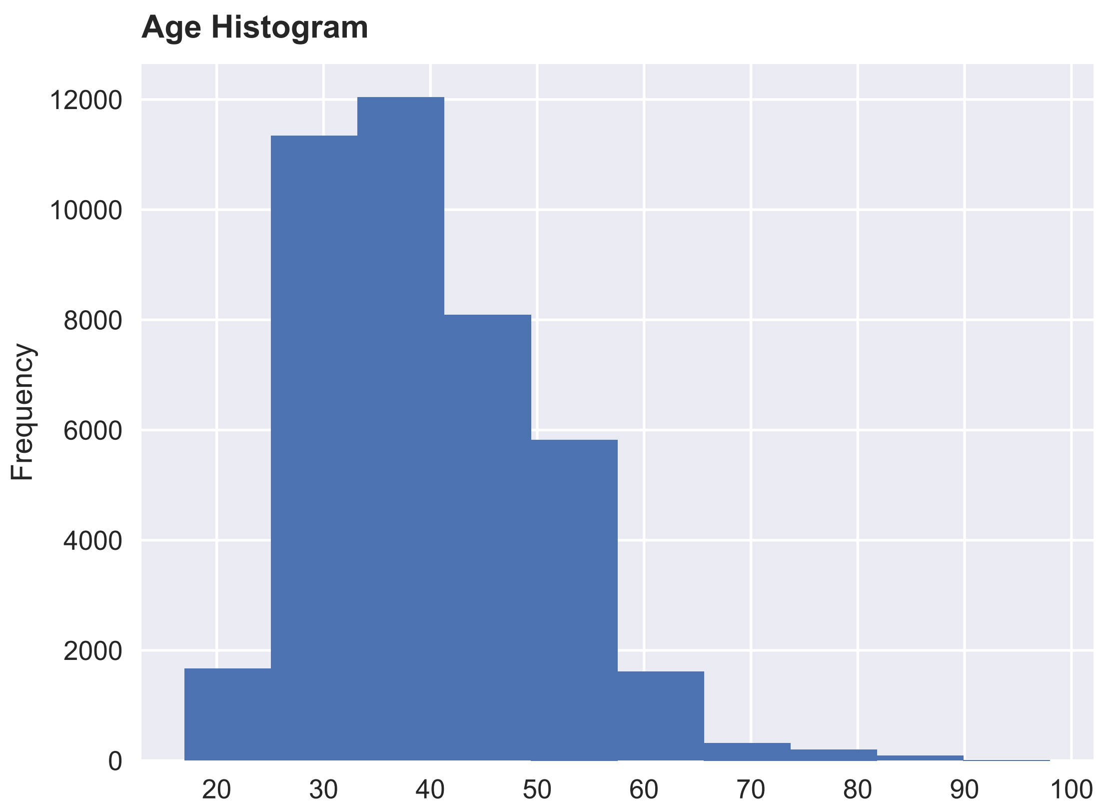


1.2.2 Client Data

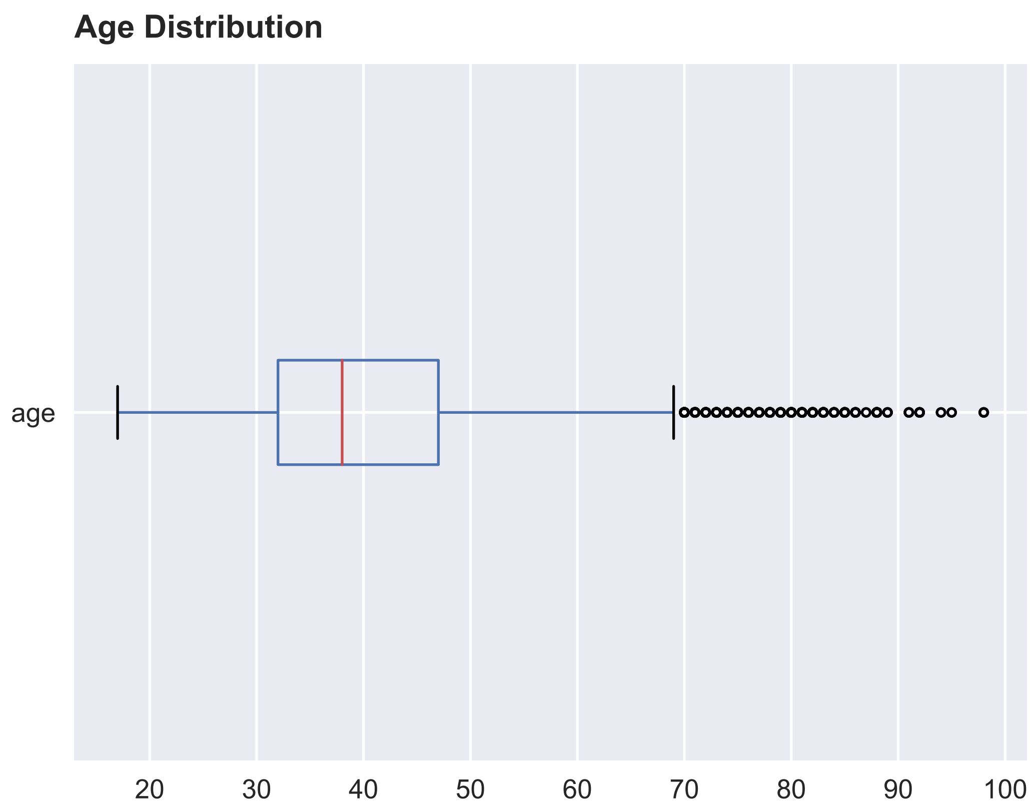
Let's start with basic client data.

Most of the clients's age are between 32 to 47 while there are some outlier cases beyond 70. This may imply that we should choose standardization for scaling since it's more tolerant for outliers.

[10]:
```
age_hist = bank_mkt["age"].plot.hist(title="Age Histogram")
```

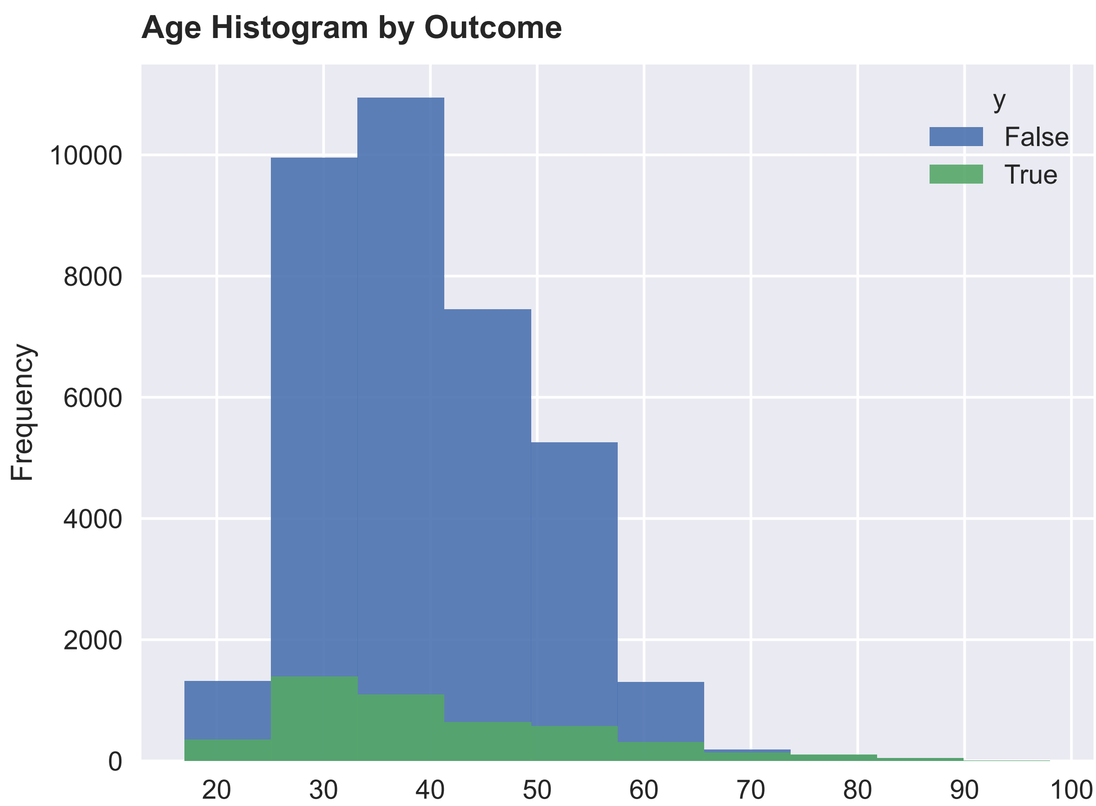


[11]:
```
age_box = bank_mkt["age"].plot.box(vert=False, sym=".", title="Age␣
↪Distribution")
```

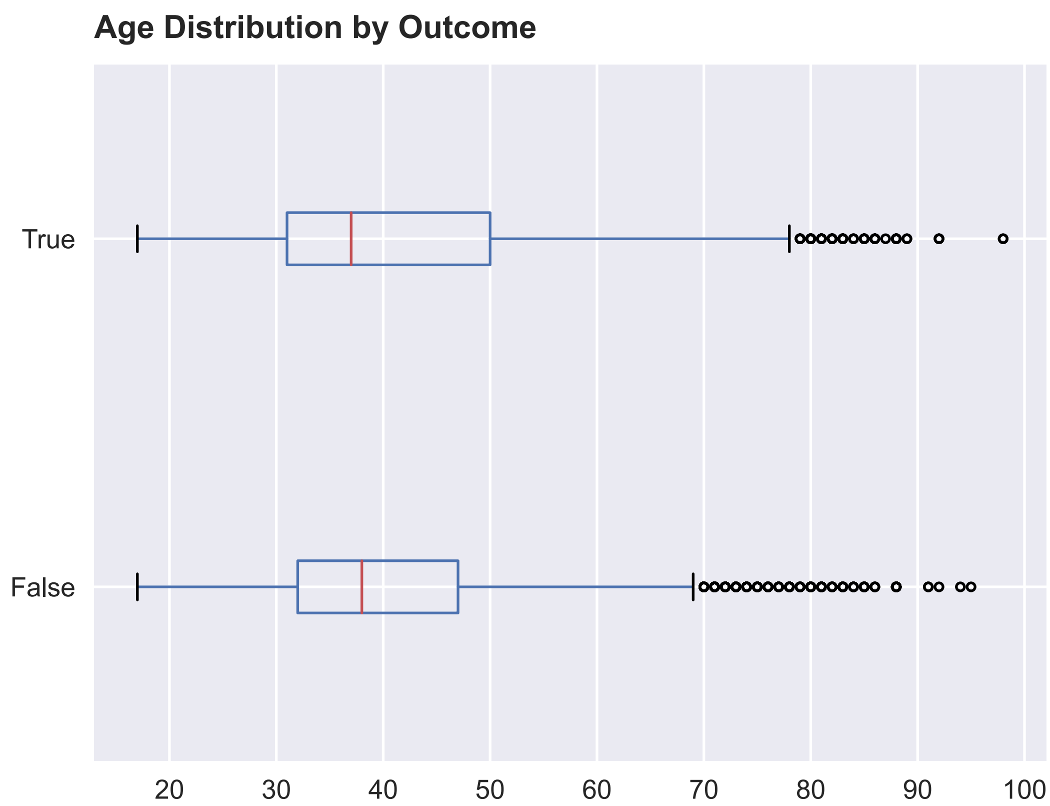


From the graph below we can see that the age distribution in the true outcome group has lower median age but is more skewed toward an slightly older population.

[12]:
```
age_y = bank_mkt[["age", "y"]].pivot(columns="y", values="age")
age_hist_outcome = age_y.plot.hist(alpha=0.9, legend=True, title="Age Histogram␣
↪by Outcome")
```



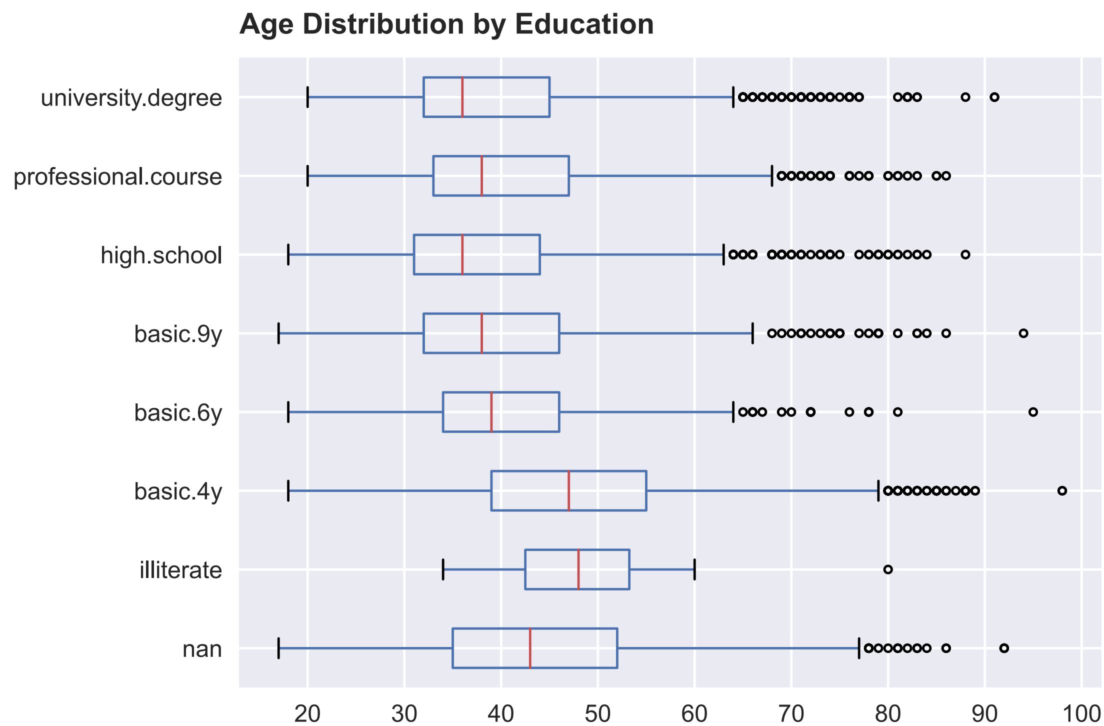

```
[13]: age_box_outcome = age_y.plot.box(vert=False, sym=".", title="Age Distribution␣
      ↪by Outcome")
```



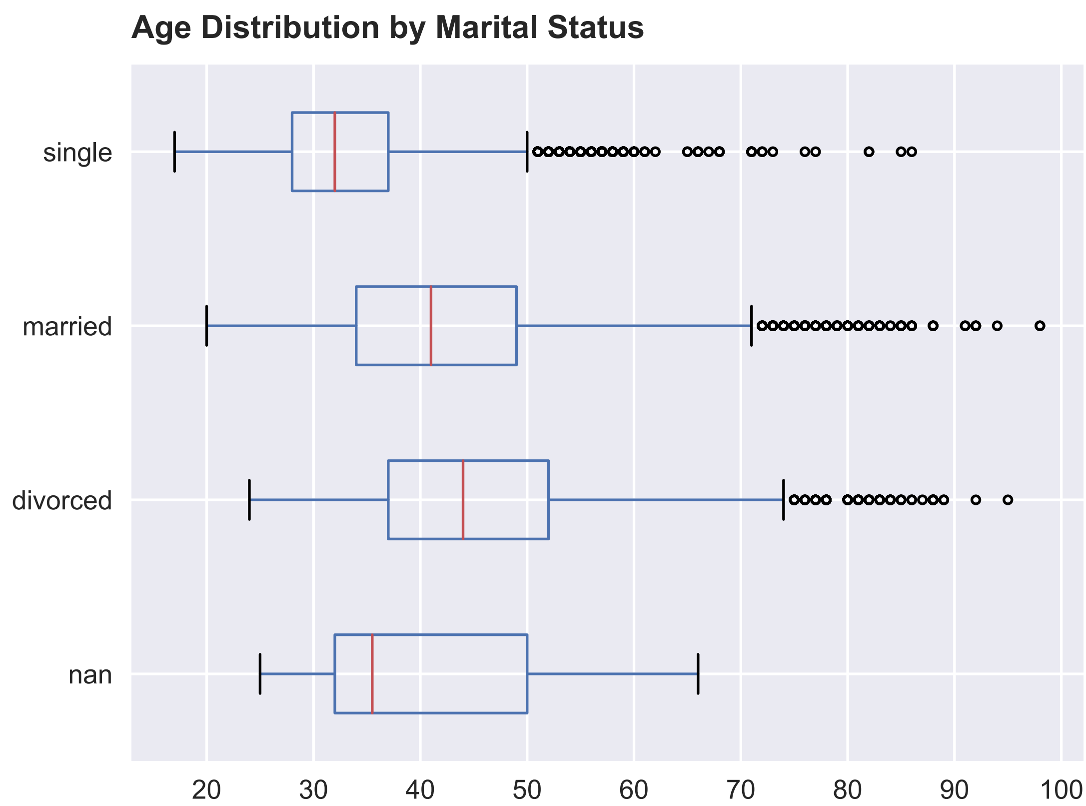
We can also inspect the relationship between age and other categorical values.

[14]:
```
age_job = bank_mkt[["age", "job"]].pivot(columns="job", values="age")
age_job_box = age_job.plot.box(vert=False, sym=".", title="Age Distribution by␣
↪Job")
```

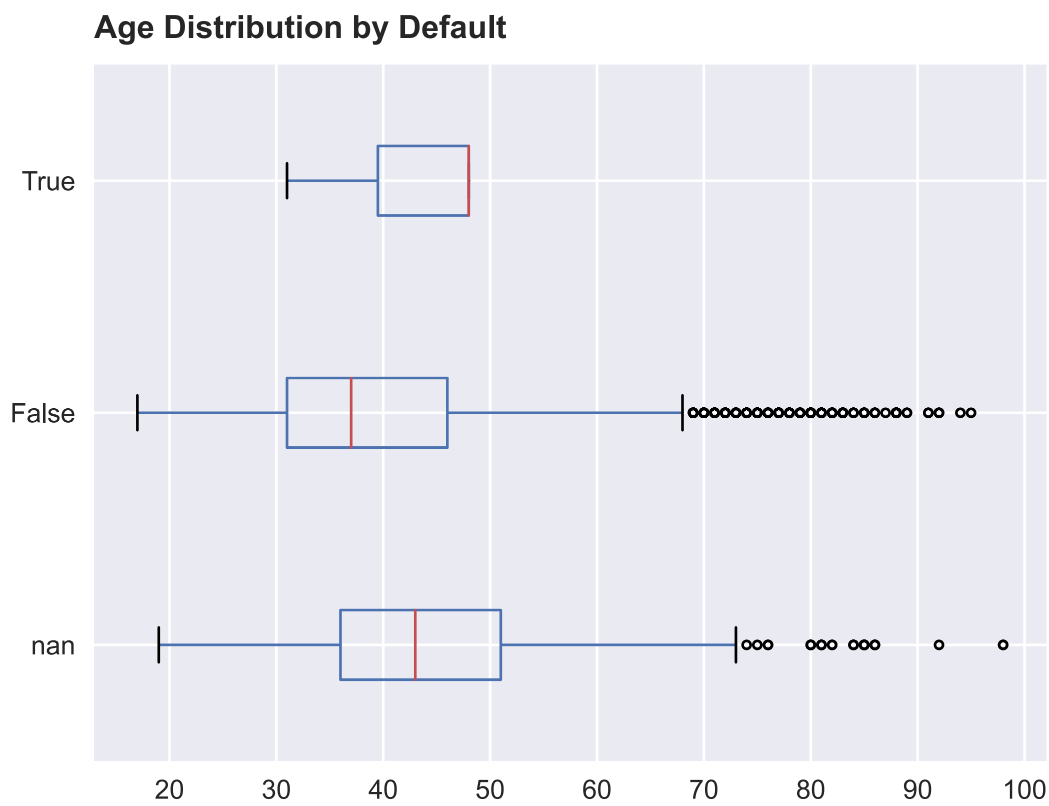
[15]:
```
age_education = bank_mkt[["age", "education"]].pivot(columns="education",␣
↪values="age")
age_education_box = age_education.plot.box(vert=False, sym=".", title="Age␣
↪Distribution by Education")
```



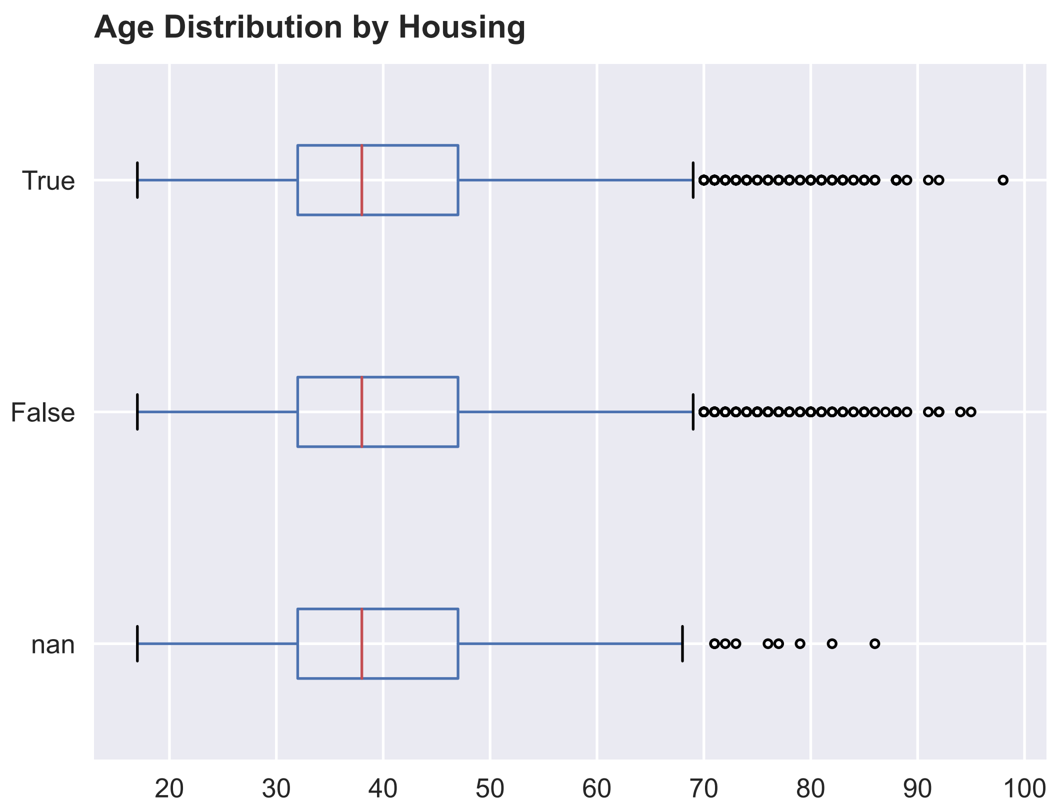
[16]:
```
age_marital = bank_mkt[["age", "marital"]].pivot(columns="marital",␣
↪values="age")
age_marital_box = age_marital.plot.box(vert=False, sym=".", title="Age␣
↪Distribution by Marital Status")
```



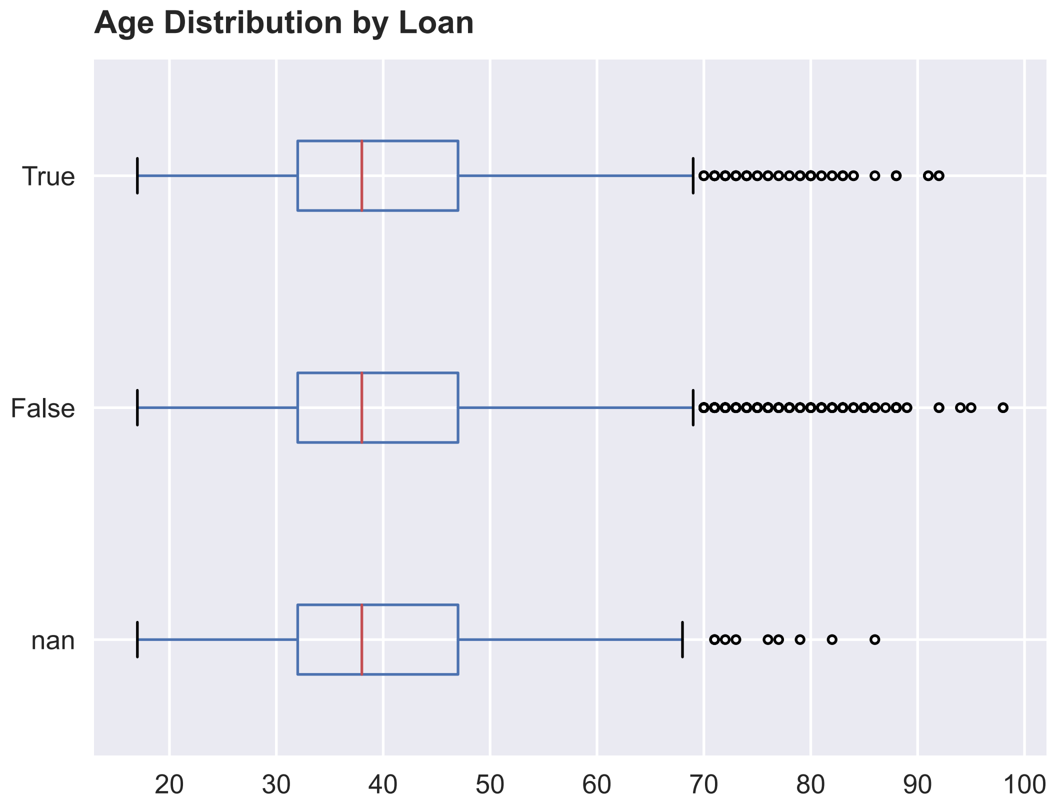
[17]:
```
age_default = bank_mkt[["age", "default"]].pivot(columns="default",␣
↪values="age")
age_default_box = age_default.plot.box(vert=False, sym=".", title="Age␣
↪Distribution by Default")
```



[18]:
```
age_housing = bank_mkt[["age", "housing"]].pivot(columns="housing",␣
↪values="age")
age_housing_box = age_housing.plot.box(vert=False, sym=".", title="Age␣
↪Distribution by Housing")
```



[19]:
```
age_loan = bank_mkt[["age", "loan"]].pivot(columns="loan", values="age")
age_loan_box = age_loan.plot.box(vert=False, sym=".", title="Age Distribution␣
↪by Loan")
```



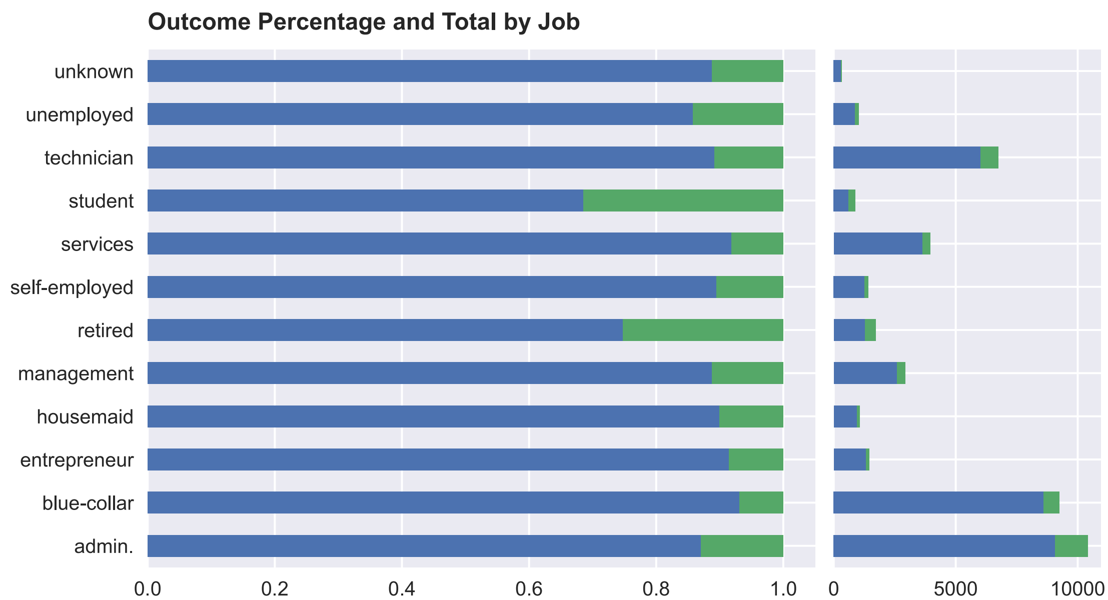
We can then turn to job, eductaion and other categorical data to see their relationship to the outcome.

[20]:
```
def explore_cat(df, feature):
    df = df.copy()
    if pd.api.types.is_categorical_dtype(df[feature]):
        df[feature] = df[feature].cat.add_categories('unknown')
        df[feature] = df[feature].fillna("unknown")
    feature_true = df[[feature, "y"]].groupby([feature]).sum().y.rename("True")
    feature_total = df[[feature, "y"]].groupby([feature]).count().y.
 ↪rename("Total")
    feature_false = feature_total - feature_true
    feature_false = feature_false.rename("False")
    feature_true_rate = feature_true / feature_total
    feature_true_rate = feature_true_rate.rename("True Percentage")
    explore_df = pd.concat([feature_true, feature_false, feature_total,␣
 ↪feature_true_rate], axis=1).reset_index()
    return explore_df
```

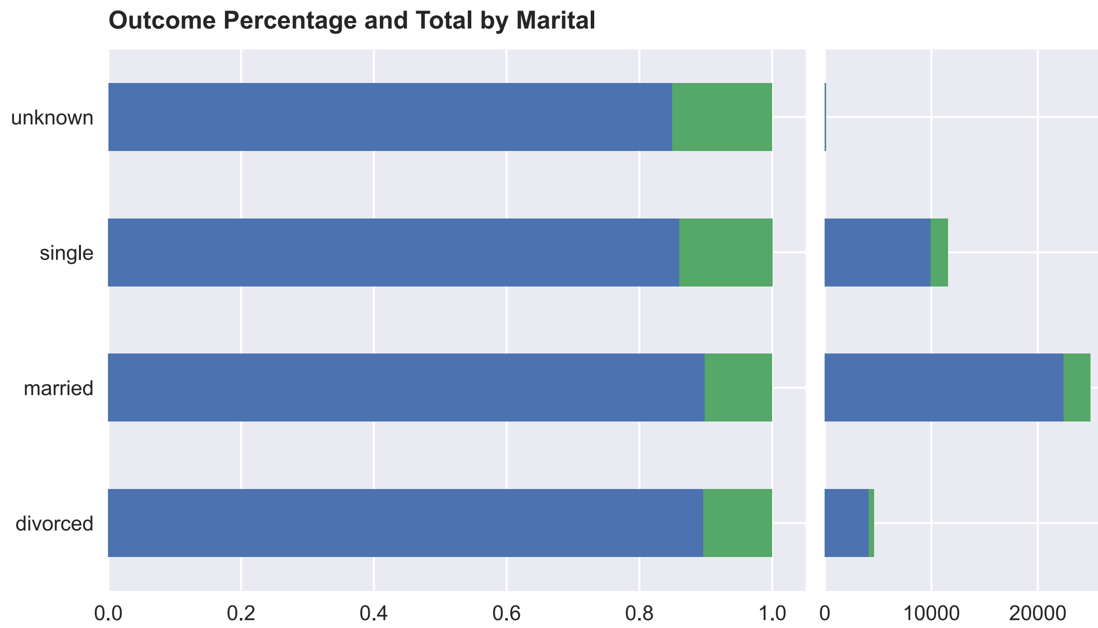
[21]:
```
def cat_outcome(df, feature):
    df = df.copy()
```

```
    if pd.api.types.is_categorical_dtype(df[feature]) and df[feature].isna().
↪sum() > 0:
        df[feature] = df[feature].cat.add_categories("unknown")
        df[feature] = df[feature].fillna("unknown")
    title = feature.title().replace("_", " ").replace("Of", "of")
    f, axs = plt.subplots(1, 2, figsize=(8.6, 4.8), sharey=True,␣
↪gridspec_kw=dict(wspace=0.04, width_ratios=[5, 2]))
    ax0 = df["y"].groupby(df[feature], dropna=False).
↪value_counts(normalize=True).unstack().plot.barh(xlabel="", legend=False,␣
↪stacked=True, ax=axs[0], title=f"Outcome Percentage and Total by {title}")
    ax1 = df["y"].groupby(df[feature], dropna=False).value_counts().unstack().
↪plot.barh(xlabel="", legend=False, stacked=True, ax=axs[1])
```

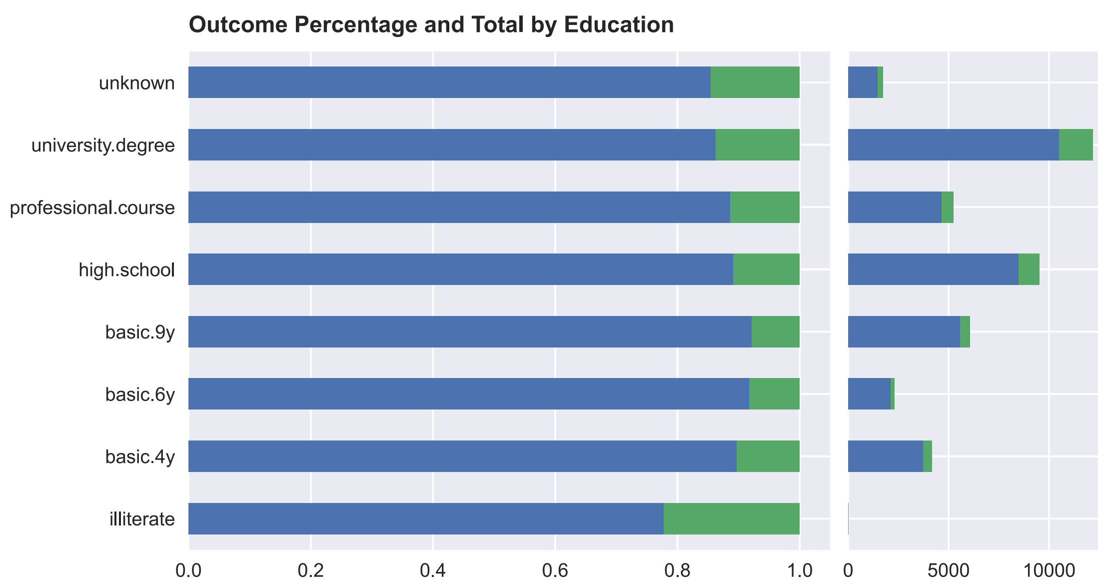
[22]:
```
job_outcome = cat_outcome(bank_mkt, "job")
```



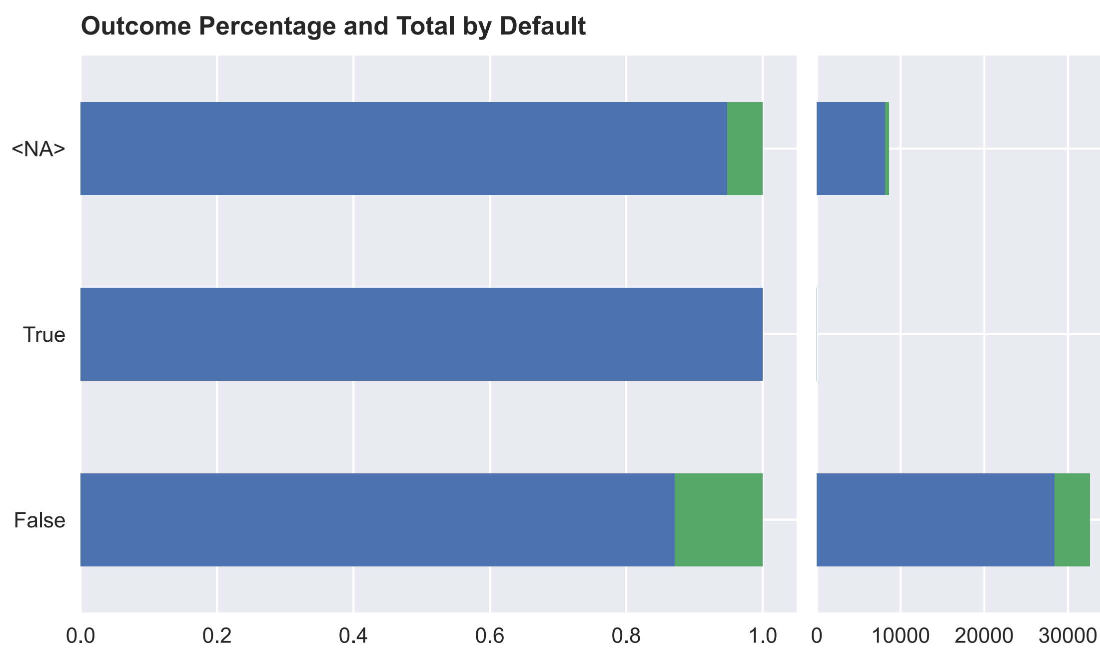
[23]:
```
marital_outcome = cat_outcome(bank_mkt, "marital")
```



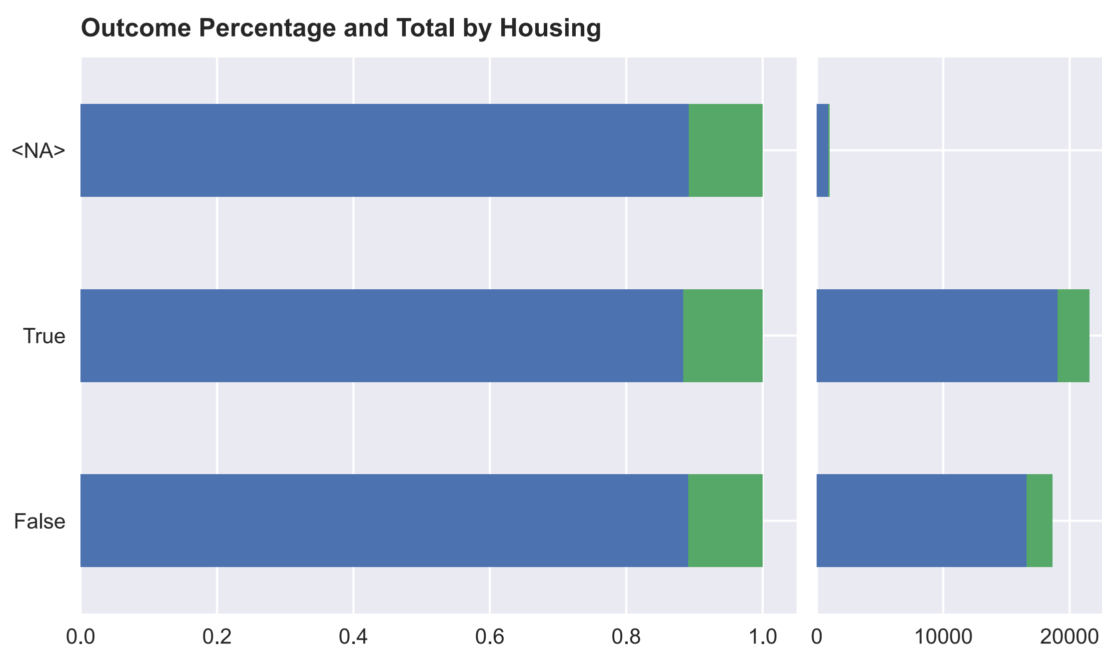
[24]:
```
education_outcome = cat_outcome(bank_mkt, "education")
```



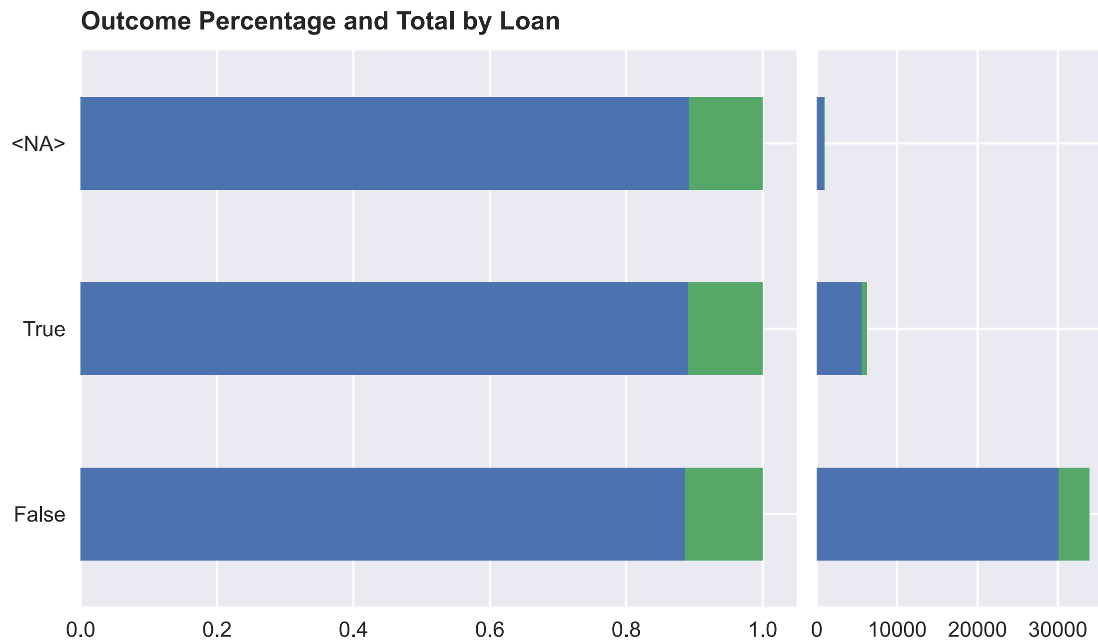
[25]:
```
default_outcome = cat_outcome(bank_mkt, "default")
```



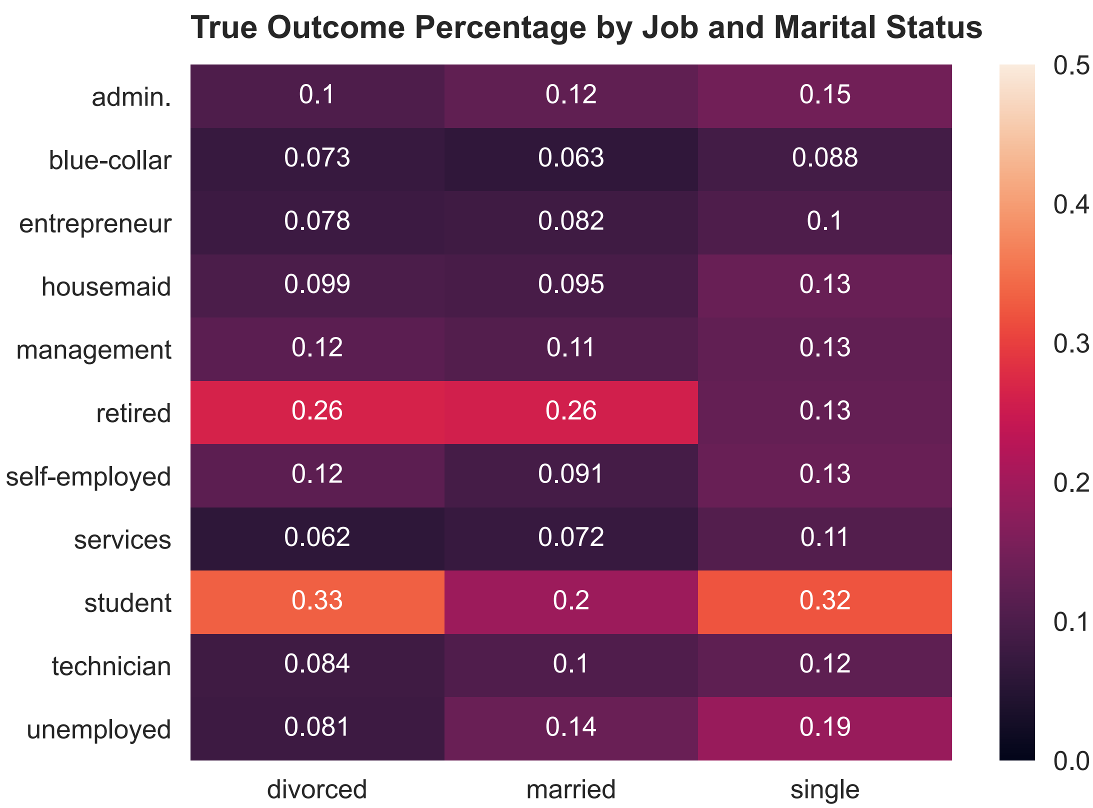
[26]:
```
housing_outcome = cat_outcome(bank_mkt, "housing")
```



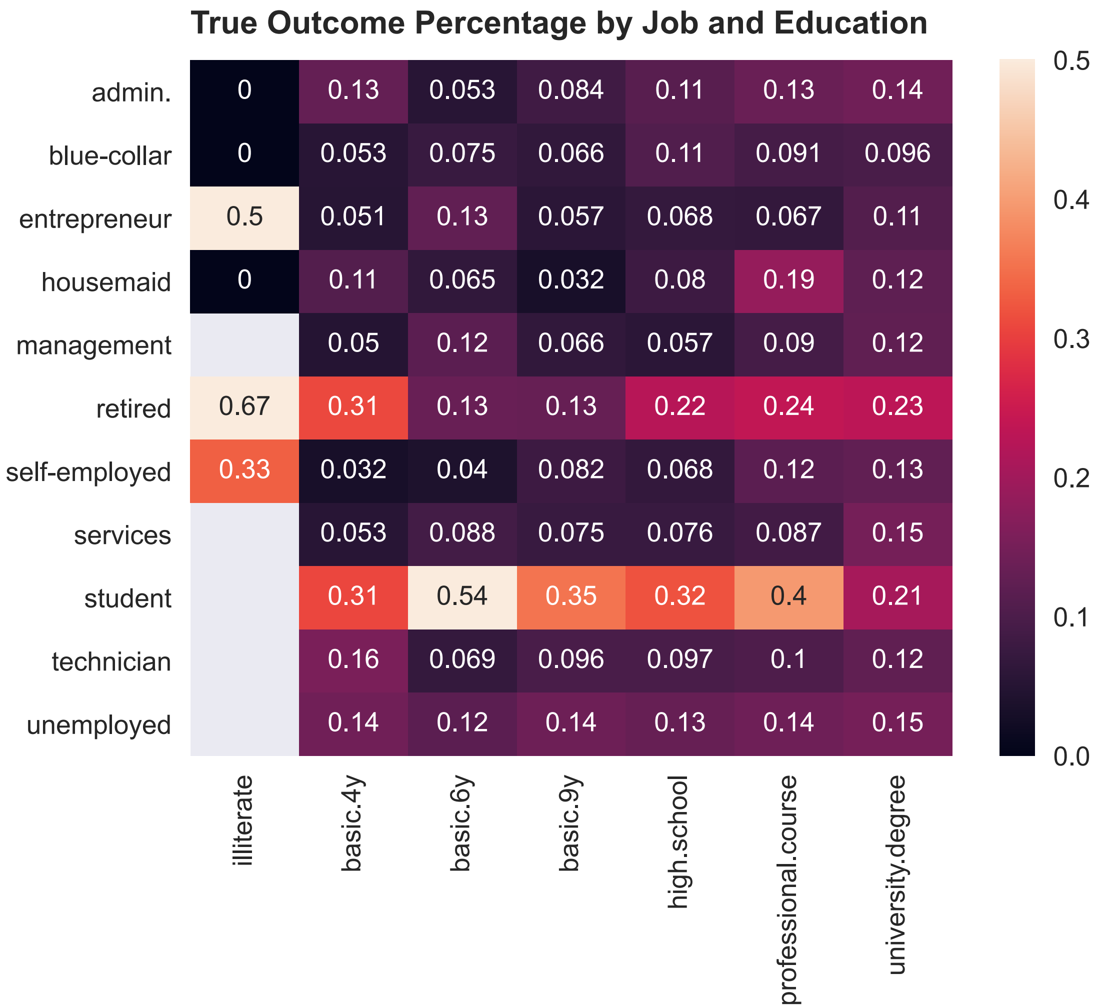
[27]:
```
loan_outcome = cat_outcome(bank_mkt, "loan")
```



[28]:
```
job_marital_total = bank_mkt[["job", "marital", "y"]].groupby(["job",␣
↪"marital"]).count().y.unstack()
job_marital_true = bank_mkt[["job", "marital", "y"]].groupby(["job",␣
↪"marital"]).sum().y.unstack()
job_marital_rate = job_marital_true / job_marital_total
job_marital_rate = job_marital_rate.rename_axis(None, axis=0).rename_axis(None,␣
↪axis=1)
job_marital_heatmap = sns.heatmap(data=job_marital_rate, vmin=0, vmax=0.5,␣
↪annot=True).set_title("True Outcome Percentage by Job and Marital Status")
```

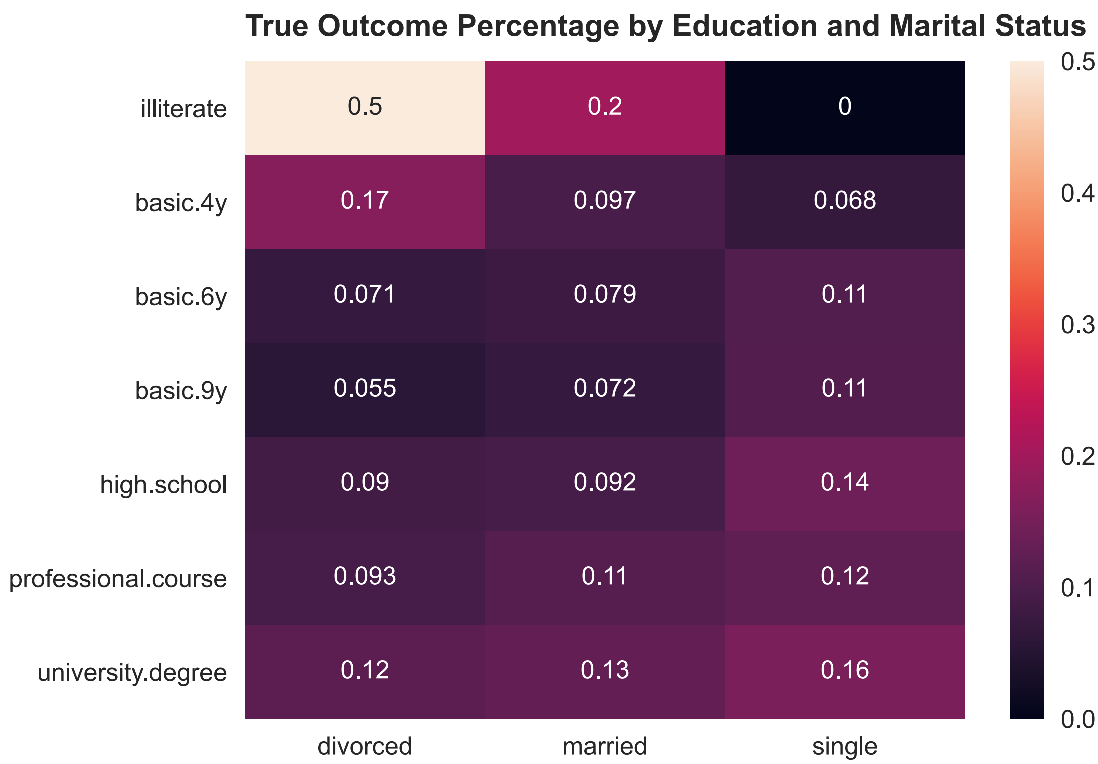


[29]:
```
job_education_total = bank_mkt[["job", "education", "y"]].groupby(["job",␣
 ↪"education"]).count().y.unstack()
job_education_true = bank_mkt[["job", "education", "y"]].groupby(["job",␣
 ↪"education"]).sum().y.unstack()
job_education_rate = job_education_true / job_education_total
job_education_rate = job_education_rate.rename_axis(None, axis=0).
 ↪rename_axis(None, axis=1)
job_education_heatmap = sns.heatmap(data=job_education_rate, vmin=0, vmax=0.5,␣
 ↪annot=True).set_title("True Outcome Percentage by Job and Education")
```



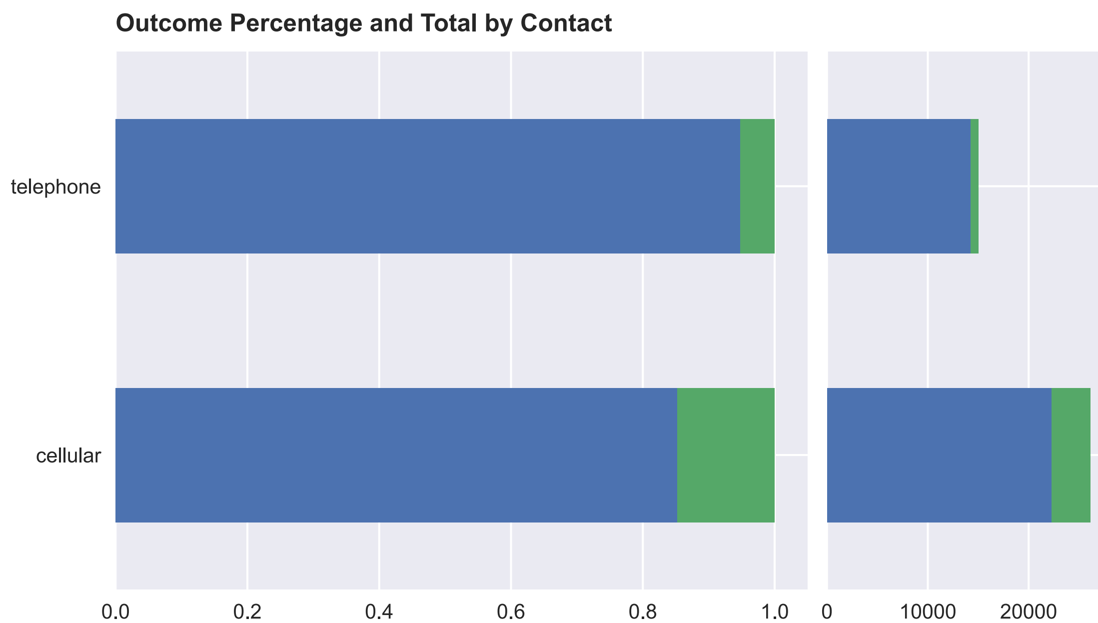
[30]:

```
education_marital_total = bank_mkt[["education", "marital", "y"]].
↪groupby(["education", "marital"]).count().y.unstack()
education_marital_true = bank_mkt[["education", "marital", "y"]].
↪groupby(["education", "marital"]).sum().y.unstack()
education_marital_rate = education_marital_true / education_marital_total
education_marital_rate = education_marital_rate.rename_axis(None, axis=0).
↪rename_axis(None, axis=1)
education_marital_heatmap = sns.heatmap(data=education_marital_rate, vmin=0,␣
↪vmax=0.5, annot=True).set_title("True Outcome Percentage by Education and␣
↪Marital Status")
```

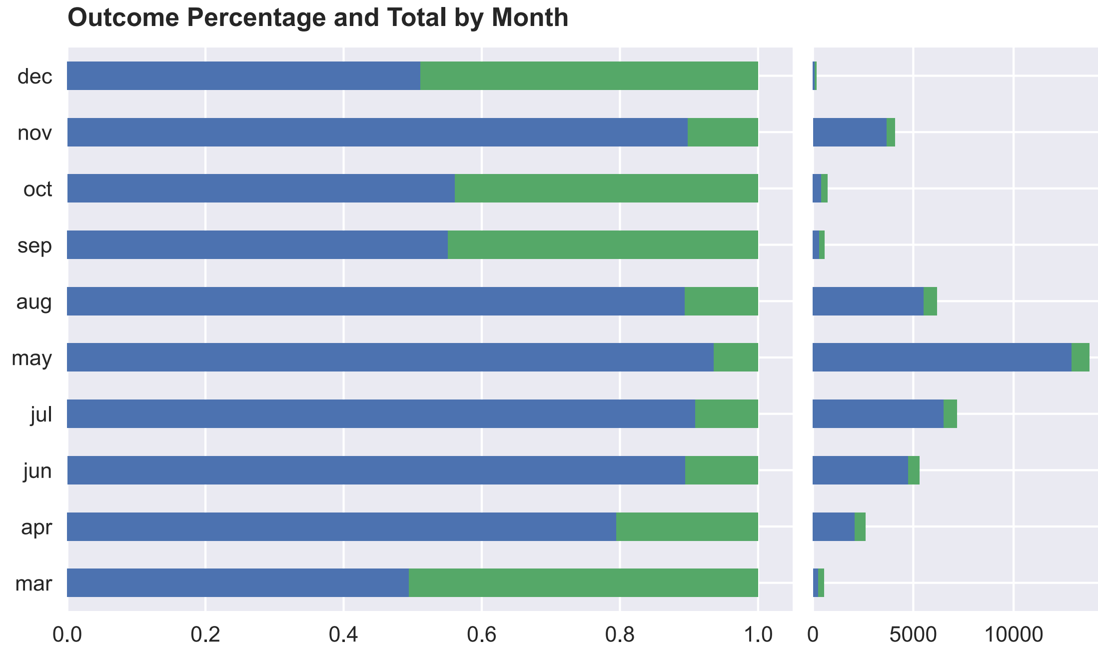


1.2.3 Current Campaign

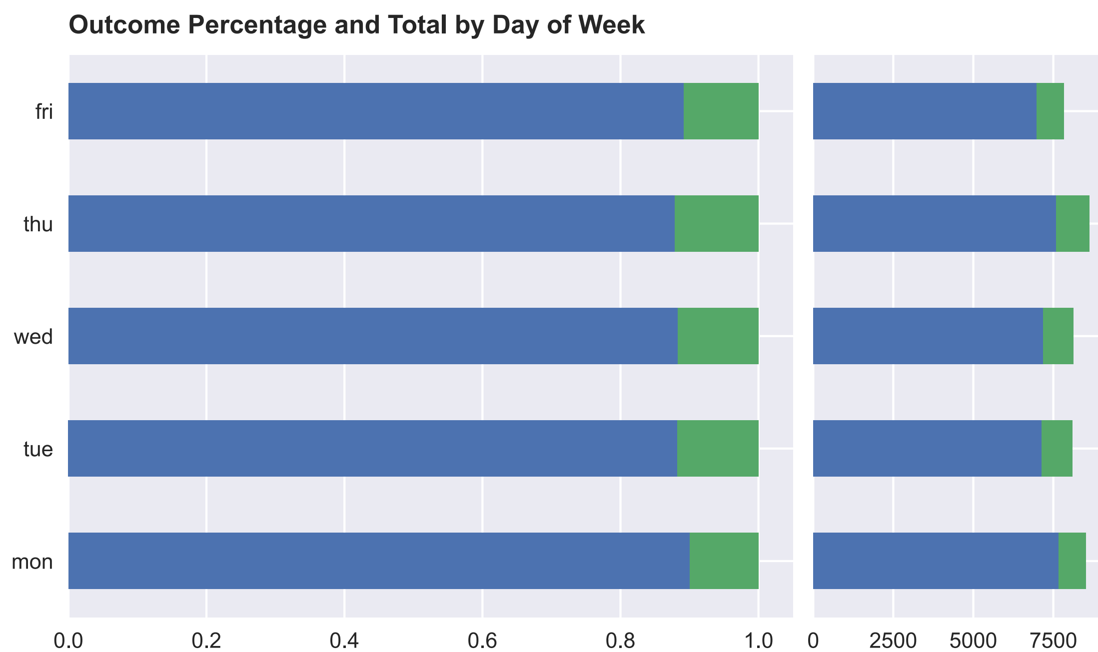
[31]:
```
contact_outcome = cat_outcome(bank_mkt, "contact")
```



[32]:
```
month_outcome = cat_outcome(bank_mkt, "month")
```



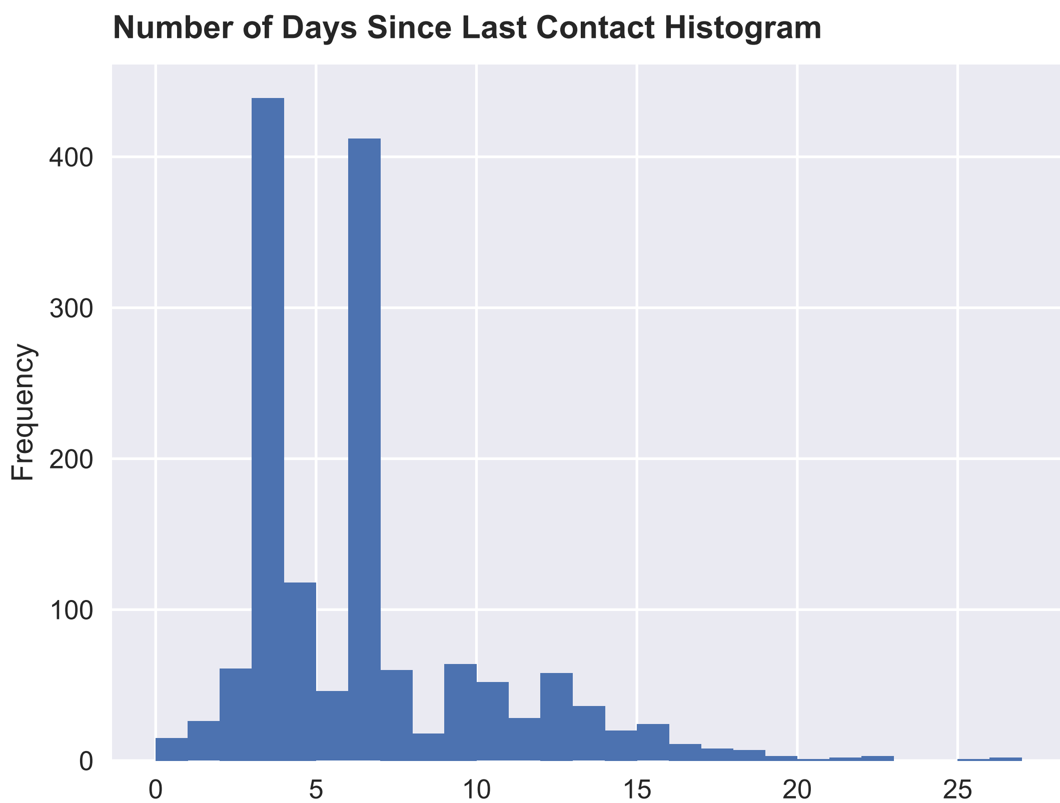
[33]:
```
day_outcome = cat_outcome(bank_mkt, "day_of_week")
```



1.2.4 Previous Campaign

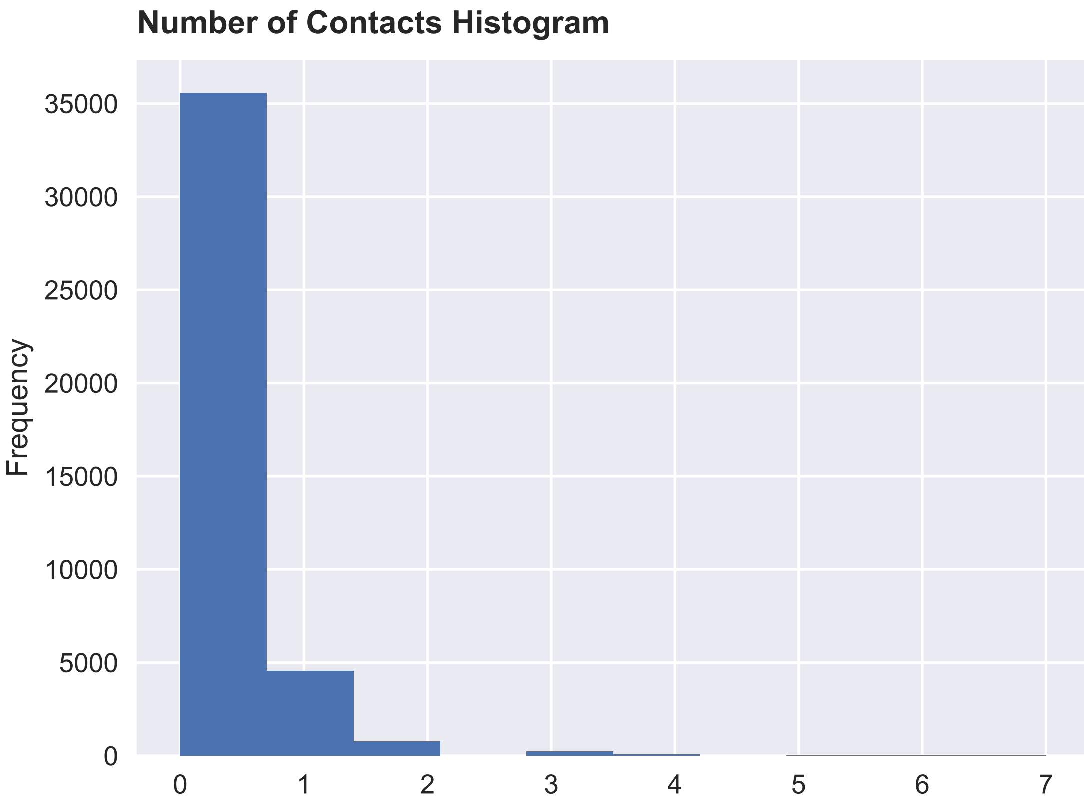
We can plot the dirstribution of `pdays` and `previous`. As we can see, most of the client with `pdays` has been contacted 3 to 6 days before and peaked at 3 and 6 days.

[34]:
```
pdays_hist = bank_mkt["pdays"].plot.hist(bins=27, title="Number of Days Since␣
↪Last Contact Histogram")
```



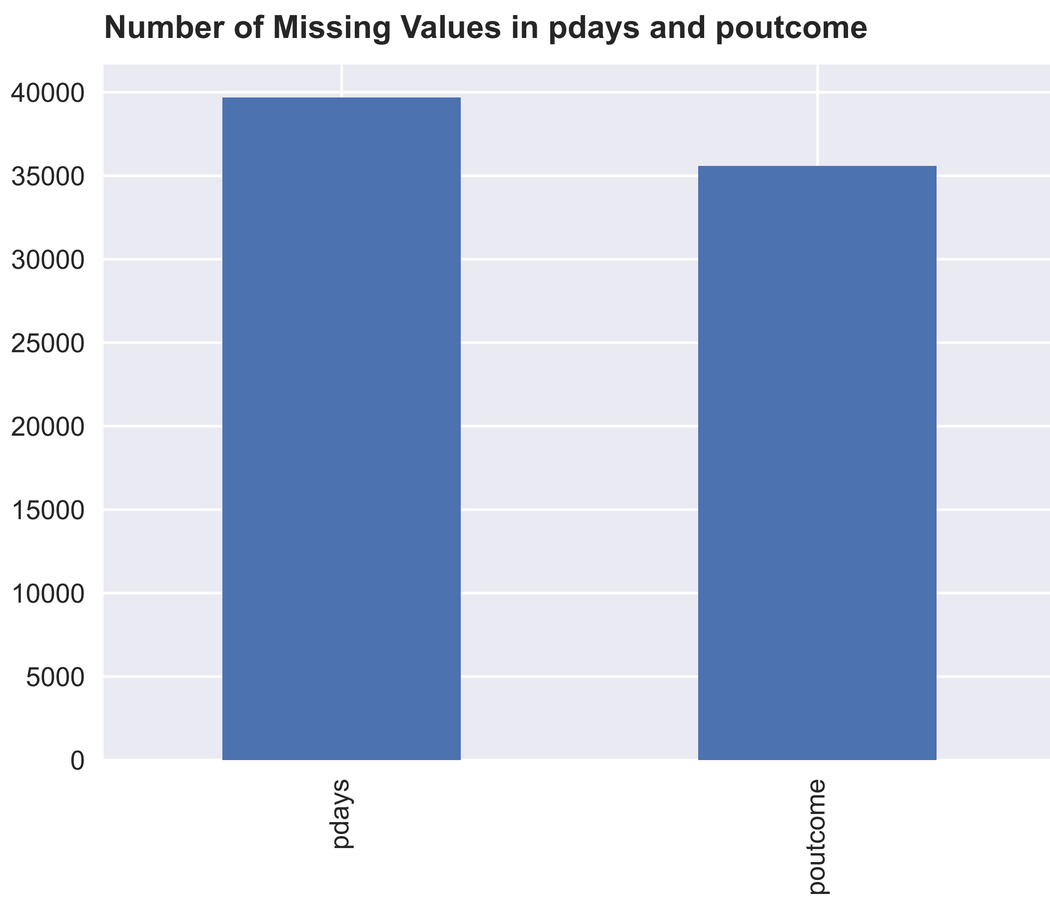
Most of the client has never been contacted before.

[35]:
```
previous_hist = bank_mkt["previous"].plot.hist(title="Number of Contacts␣
↪Histogram")
```



If `pdays` is missing value, that means that the client was not previously contacted and therefore should not have `poutcome`. But `poutcome` column has less missing values than `pdays`.

[36]:
```
previous_na = bank_mkt[["pdays", "poutcome"]].isna().sum()
previous_na_ax = previous_na.plot.bar(title="Number of Missing Values in pdays␣
↪and poutcome")
```



We can print out the 4110 rows where the client is not contacted but have `poutcome` and see how many times they have been contacted before. The figures suggest that maybe these clients has been actually contacted but it was more than 30 days ago so the contact date was not recorded. This leaves us plenty room for feature engineering.

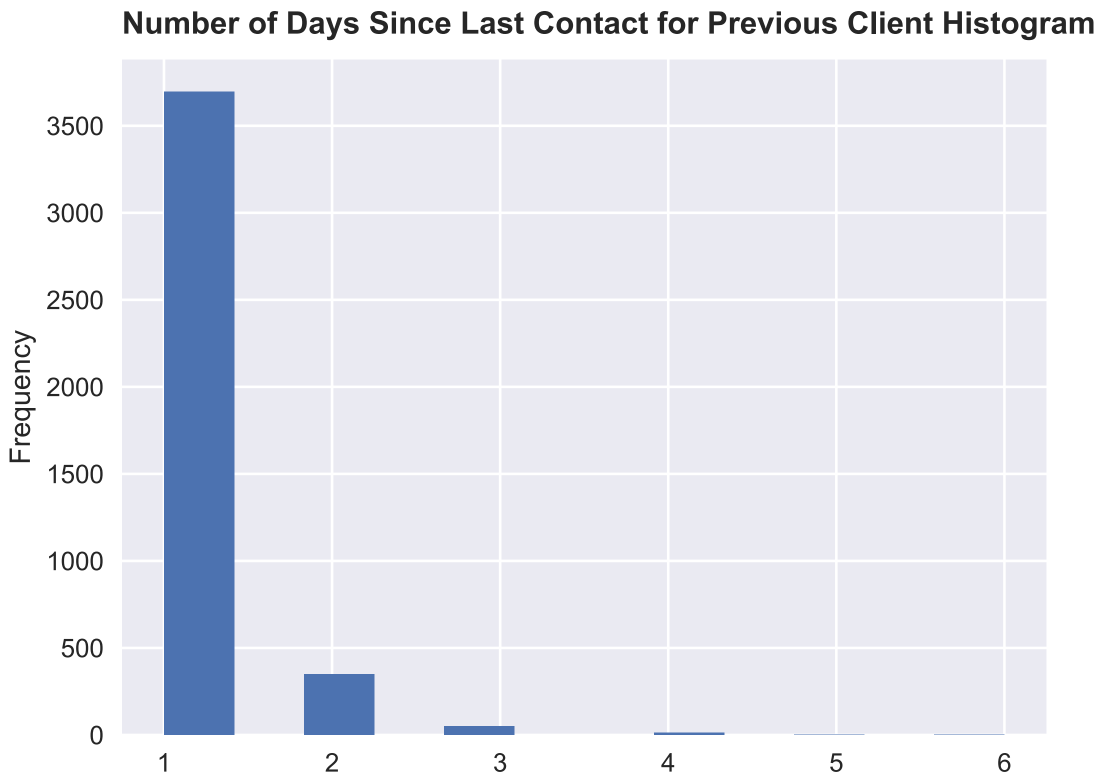
[37]:
```
previous = bank_mkt[["campaign", "pdays", "previous", "poutcome", "y"]]
previous = previous[previous["pdays"].isna() & previous["poutcome"].notna()].
 ↪reset_index(drop=True)
previous
```

[37]:
```
      campaign  pdays  previous  poutcome      y
0            1   <NA>         1     False  False
1            1   <NA>         1     False   True
2            1   <NA>         1     False  False
3            1   <NA>         1     False   True
4            1   <NA>         1     False  False
…          …     …         …       …      …
4105         1   <NA>         1     False   True
4106         2   <NA>         4     False  False
```

```
4107          1   <NA>         2     False   True
4108          1   <NA>         2     False  False
4109          3   <NA>         1     False  False

[4110 rows x 5 columns]
```

[38]:
```
previous_ax = previous["previous"].plot.hist(bins=12, title="Number of Days␣
 ↪Since Last Contact for Previous Client Histogram")
```



[39]:
```
bank_mkt[bank_mkt["pdays"].isna() & bank_mkt["poutcome"].isna()]
```

[39]:
```
       age          job  marital            education  default  housing  \
0       56    housemaid  married             basic.4y    False    False
1       57     services  married          high.school     <NA>    False
2       37     services  married          high.school    False     True
3       40       admin.  married             basic.6y    False    False
4       56     services  married          high.school    False    False
…     …           …        …                   …       …        …
41181   37       admin.  married    university.degree    False     True
41183   73      retired  married  professional.course    False     True
41184   46  blue-collar  married  professional.course    False    False
41185   56      retired  married    university.degree    False     True
```

```
41186   44   technician  married  professional.course    False    False

        loan    contact month day_of_week  …  campaign  pdays  previous  \
0      False  telephone   may         mon  …         1   <NA>         0
1      False  telephone   may         mon  …         1   <NA>         0
2      False  telephone   may         mon  …         1   <NA>         0
3      False  telephone   may         mon  …         1   <NA>         0
4       True  telephone   may         mon  …         1   <NA>         0
…        …         …    …           …   …        …     …        …
41181  False   cellular   nov         fri  …         1   <NA>         0
41183  False   cellular   nov         fri  …         1   <NA>         0
41184  False   cellular   nov         fri  …         1   <NA>         0
41185  False   cellular   nov         fri  …         2   <NA>         0
41186  False   cellular   nov         fri  …         1   <NA>         0

       poutcome  emp.var.rate  cons.price.idx  cons.conf.idx  euribor3m  \
0          <NA>           1.1          93.994          -36.4      4.857
1          <NA>           1.1          93.994          -36.4      4.857
2          <NA>           1.1          93.994          -36.4      4.857
3          <NA>           1.1          93.994          -36.4      4.857
4          <NA>           1.1          93.994          -36.4      4.857
…         …           …              …              …         …
41181      <NA>          -1.1          94.767          -50.8      1.028
41183      <NA>          -1.1          94.767          -50.8      1.028
41184      <NA>          -1.1          94.767          -50.8      1.028
41185      <NA>          -1.1          94.767          -50.8      1.028
41186      <NA>          -1.1          94.767          -50.8      1.028

       nr.employed      y
0           5191.0  False
1           5191.0  False
2           5191.0  False
3           5191.0  False
4           5191.0  False
…              …    …
41181       4963.6   True
41183       4963.6   True
41184       4963.6  False
41185       4963.6  False
41186       4963.6   True

[35563 rows x 21 columns]
```

1.2.5 Correlation Heatmap

[40]:
```
corr_heatmap = sns.heatmap(data=bank_mkt.corr(method="pearson")).
↪set_title("Correlation Heatmap")
```