WHAT DO WE GOTTA SOLVE ANYWAY?

Visualization and develop a predictive model that determines the likelihood of a startup being either acquired or closed, based on various factors such as the startup's domain, funding history, industry category, and other relevant features.

The sign "#" is a comment, so it won't be executed. If you don't understand a line just read the comment!

```
import pandas as pd #importing pandas as pd (a short name for pandas, so we don't have to ty
import matplotlib.pyplot as plt #same thing but now with matplot
import seaborn as sb #for better visuals
```

NOW LET'S IMPORT OUR DATA SET!!!!!!!!!!!!!!!

```
df = pd.read_csv('start.csv') #df means dataframe, which we gotta import from the csv file t
```

```
pd.set_option('display.max_columns', None) # just seeing all the cols
df #just prints the entire data set
```

	name	status_encoded	founded_at	closed_at	first_funding_at	last_funding_a
0	Bandsintown	1	2007-01-01	0	4/1/2009	1/1/201
1	TriCipher	1	2000-01-01	0	2/14/2005	12/28/200
2	Plixi	1	2009-03-18	0	3/30/2010	3/30/201
3	Solidcore Systems	1	2002-01-01	0	2/17/2005	4/25/200
4	Inhale Digital	0	2010-08-01	10/1/2012	8/1/2010	4/1/201
...	...	...	...	...	...	
918	CoTweet	1	2009-01-01	0	7/9/2009	7/9/200
919	Reef Point Systems	0	1998-01-01	6/25/2008	4/1/2005	3/23/200
920	Paracor Medical	0	1999-01-01	6/17/2012	6/29/2007	6/29/200
921	Causata	1	2009-01-01	0	10/5/2009	11/1/201
922	Asempra Technologies	1	2003-01-01	0	2/13/2006	2/13/200

923 rows × 39 columns

```
df.drop("Unnamed: 0", axis=1, inplace=True)
df.drop("Unnamed: 6", axis=1, inplace=True)
```

```
#deleting all the cols which I think is not needed
```

```
df.category_code
```

```
0            music
1       enterprise
2              web
3         software
4      games_video
          ...
918    advertising
919       security
920        biotech
921       software
922       security
Name: category_code, Length: 923, dtype: object
```

Now Understand the dataset and what they mean. Should have done this earlier huh!

```
age_first_funding_year - quantitative
age_last_funding_year - quantitative
relationships - quantitative
funding_rounds - quantitative
funding_total_usd - quantitative
milestones - quantitative
age_first_milestone_year - quantitative
age_last_milestone_year - quantitative
state - categorical
industry_type - categorical
has_VC - categorical
has_angel - categorical
has_roundA - categorical
has_roundB - categorical
has_roundC - categorical
has_roundD - categorical
avg_participants - quantitative
is_top500 - categorical
status(acquired/closed) - categorical (the target variable, if a startup is 'acquired' by some other
```

```
df.columns #look at the columns
```

```
Index(['state_code', 'latitude', 'longitude', 'zip_code', 'city', 'name',
       'labels', 'founded_at', 'closed_at', 'first_funding_at',
       'last_funding_at', 'age_first_funding_year', 'age_last_funding_year',
       'age_first_milestone_year', 'age_last_milestone_year', 'relationships',
       'funding_rounds', 'funding_total_usd', 'milestones', 'state_code.1',
       'is_CA', 'is_NY', 'is_MA', 'is_TX', 'is_otherstate', 'category_code',
       'is_software', 'is_web', 'is_mobile', 'is_enterprise', 'is_advertising',
       'is_gamesvideo', 'is_ecommerce', 'is_biotech', 'is_consulting',
       'is_othercategory', 'has_VC', 'has_angel', 'has_roundA', 'has_roundB',
       'has_roundC', 'has_roundD', 'avg_participants', 'is_top500', 'status'],
```

```
      dtype='object')
```

```
df.shape #Just the shape with rows and col.
```

```
(923, 45)
```

```
df.isnull().sum()
```

```
state_code                  0
latitude                    0
longitude                   0
zip_code                    0
city                        0
name                        0
labels                      0
founded_at                  0
closed_at                 588
first_funding_at            0
last_funding_at             0
age_first_funding_year      0
age_last_funding_year       0
age_first_milestone_year  152
age_last_milestone_year   152
relationships               0
funding_rounds              0
funding_total_usd           0
milestones                  0
state_code.1                1
is_CA                       0
is_NY                       0
is_MA                       0
is_TX                       0
is_otherstate               0
category_code               0
is_software                 0
is_web                      0
is_mobile                   0
is_enterprise               0
is_advertising              0
is_gamesvideo               0
is_ecommerce                0
is_biotech                  0
is_consulting               0
is_othercategory            0
has_VC                      0
has_angel                   0
has_roundA                  0
has_roundB                  0
has_roundC                  0
has_roundD                  0
avg_participants            0
is_top500                   0
status                      0
dtype: int64
```

```
df.status.value_counts()
```

```
acquired    597
closed      326
Name: status, dtype: int64
```

```
#All these cols have already been One Hotted
df.drop("state_code.1", axis=1, inplace=True)
#df.drop("state_code", axis=1, inplace=True )
df.drop("city", axis=1, inplace=True)
df.drop("latitude", axis=1, inplace=True)
df.drop("longitude", axis=1, inplace=True)
df.drop("zip_code", axis=1, inplace=True)
```

```
df.category_code.value_counts().nunique()
```

```
22
```

```
df.closed_at.fillna(0, inplace=True)
```

```
"""age_first_milestone_year    152
age_last_milestone_year     152
"""
df.age_first_funding_year.fillna(0, inplace=True)
df.age_last_milestone_year.fillna(0, inplace=True)
df.age_first_milestone_year.fillna(0, inplace=True)
```

```
df.rename(columns={'labels':"status_encoded"}, inplace=True)
```

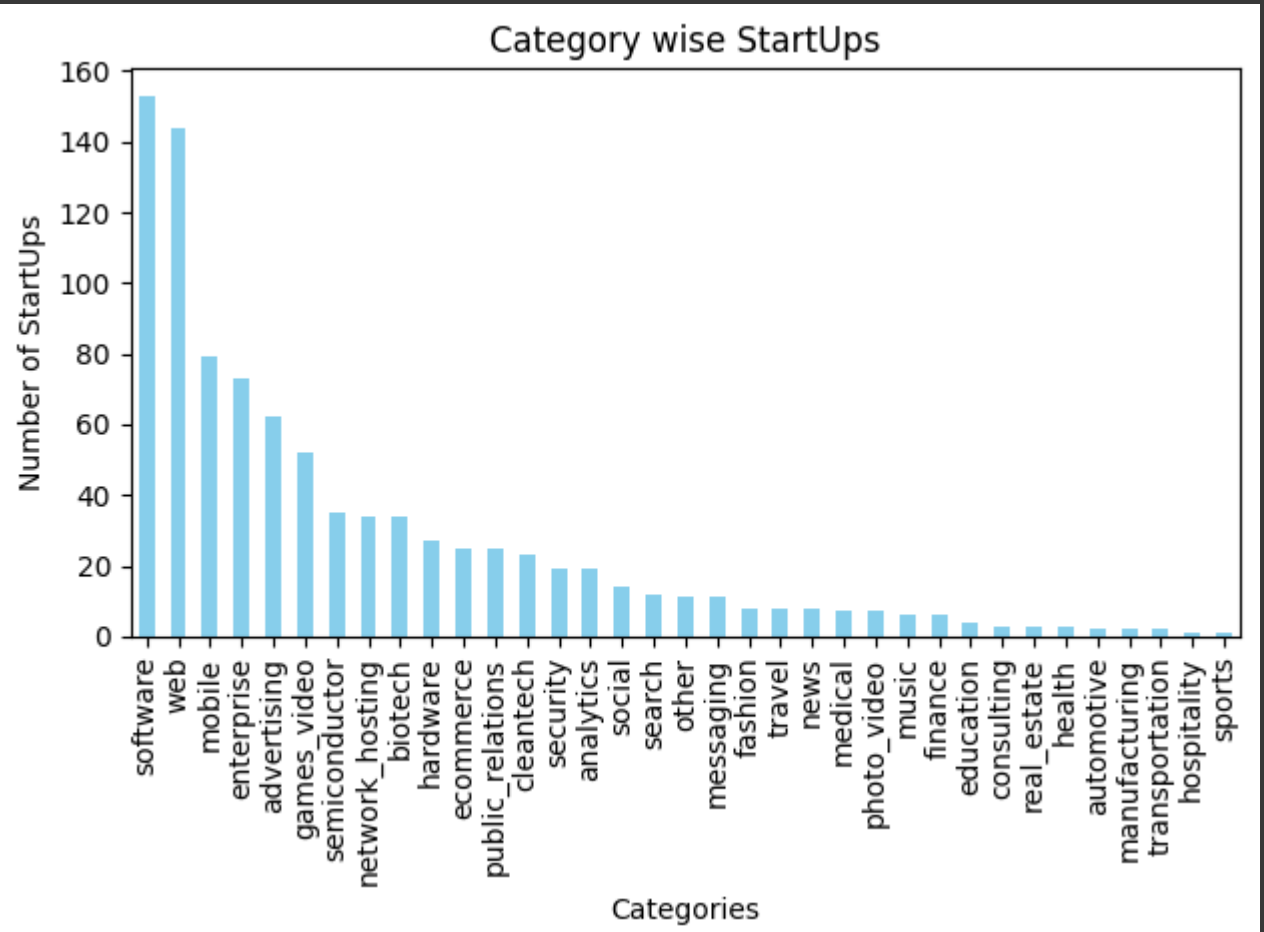
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 923 entries, 0 to 922
Data columns (total 39 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   name                     923 non-null    object
 1   status_encoded           923 non-null    int64
 2   founded_at               923 non-null    object
 3   closed_at                923 non-null    object
 4   first_funding_at         923 non-null    object
 5   last_funding_at          923 non-null    object
 6   age_first_funding_year   923 non-null    float64
 7   age_last_funding_year    923 non-null    float64
 8   age_first_milestone_year 923 non-null    float64
 9   age_last_milestone_year  923 non-null    float64
 10  relationships            923 non-null    int64
 11  funding_rounds           923 non-null    int64
 12  funding_total_usd        923 non-null    int64
 13  milestones               923 non-null    int64
 14  is_CA                    923 non-null    int64
 15  is_NY                    923 non-null    int64
 16  is_MA                    923 non-null    int64
 17  is_TX                    923 non-null    int64
 18  is_otherstate            923 non-null    int64
 19  category_code            923 non-null    object
 20  is_software              923 non-null    int64
 21  is_web                   923 non-null    int64
 22  is_mobile                923 non-null    int64
 23  is_enterprise            923 non-null    int64
 24  is_advertising           923 non-null    int64
 25  is_gamesvideo            923 non-null    int64
 26  is_ecommerce             923 non-null    int64
 27  is_biotech               923 non-null    int64
```

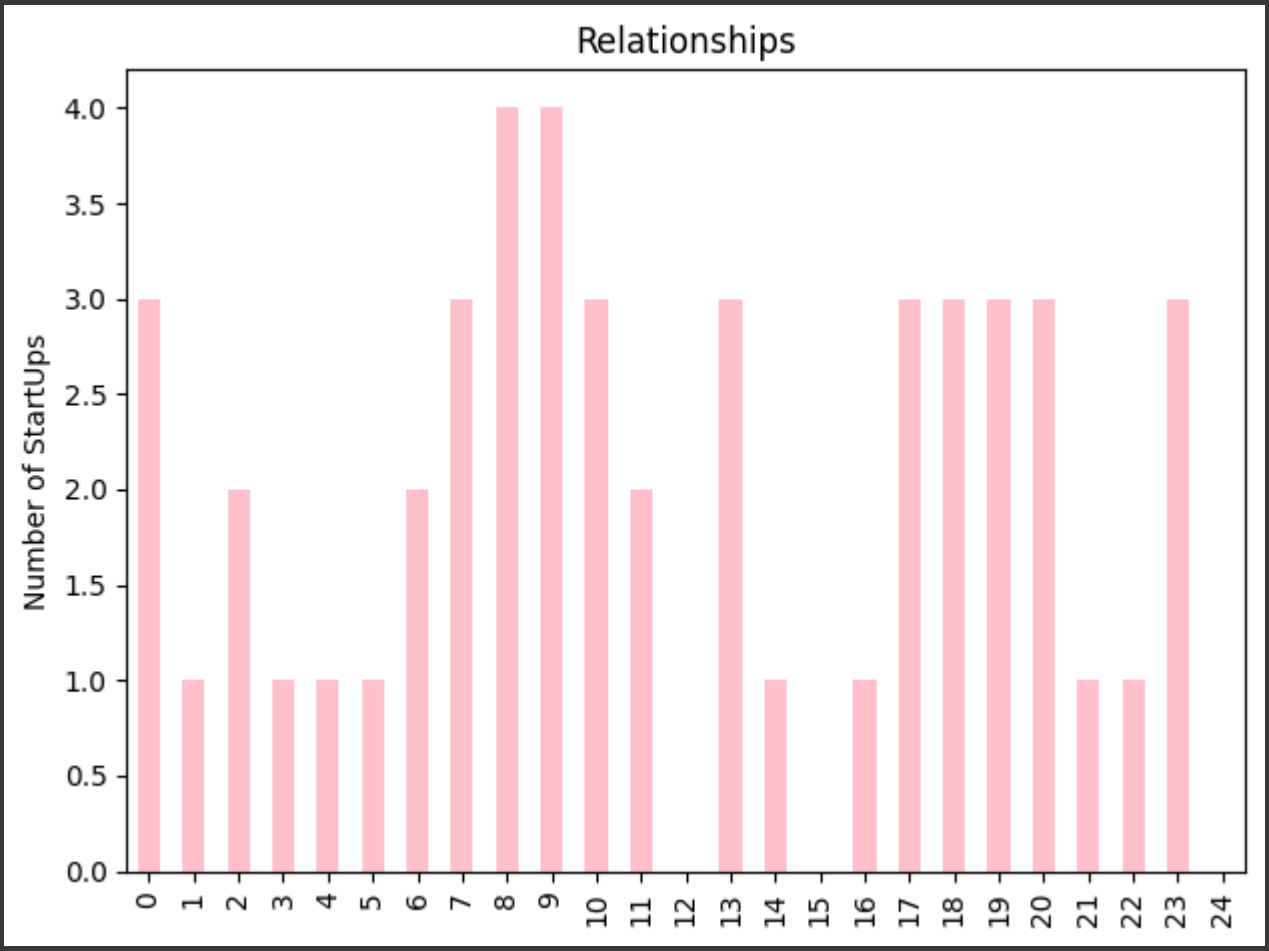
```
 28  is_consulting             923 non-null    int64
 29  is_othercategory          923 non-null    int64
 30  has_VC                    923 non-null    int64
 31  has_angel                 923 non-null    int64
 32  has_roundA                923 non-null    int64
 33  has_roundB                923 non-null    int64
 34  has_roundC                923 non-null    int64
 35  has_roundD                923 non-null    int64
 36  avg_participants          923 non-null    float64
 37  is_top500                 923 non-null    int64
 38  status                    923 non-null    object
dtypes: float64(5), int64(27), object(7)
memory usage: 281.4+ KB
```

Do visualization and develop a predictive model that determines the likelihood of a startup being either acquired or closed, based on various factors such as the startup's domain, funding history, industry category, and other relevant features.

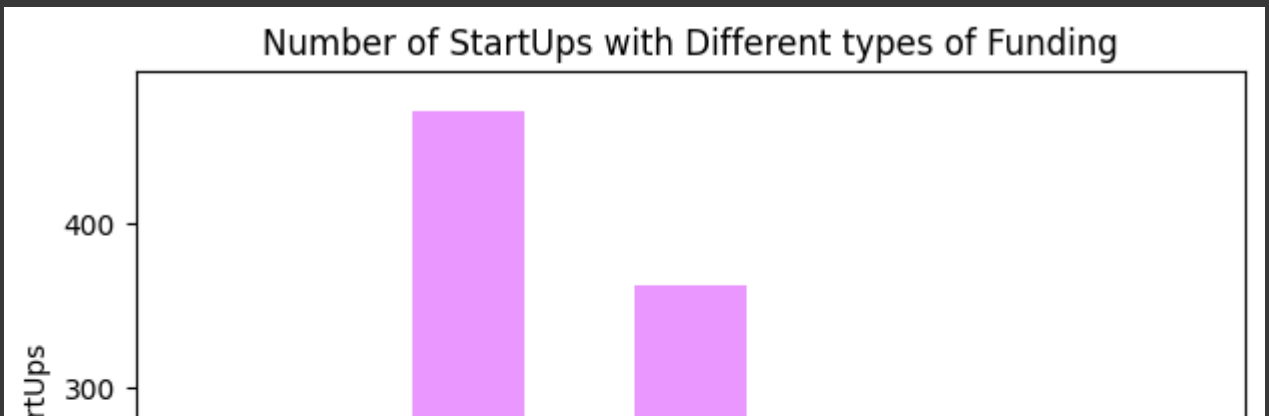
```
#finding all the category of startups
category_counts = df['category_code'].value_counts()
top_categories = category_counts
top_categories.plot(kind='bar', color='skyblue')
plt.xlabel('Categories')
plt.ylabel('Number of StartUps')
plt.title('Category wise StartUps')
plt.tight_layout()
plt.show()
```

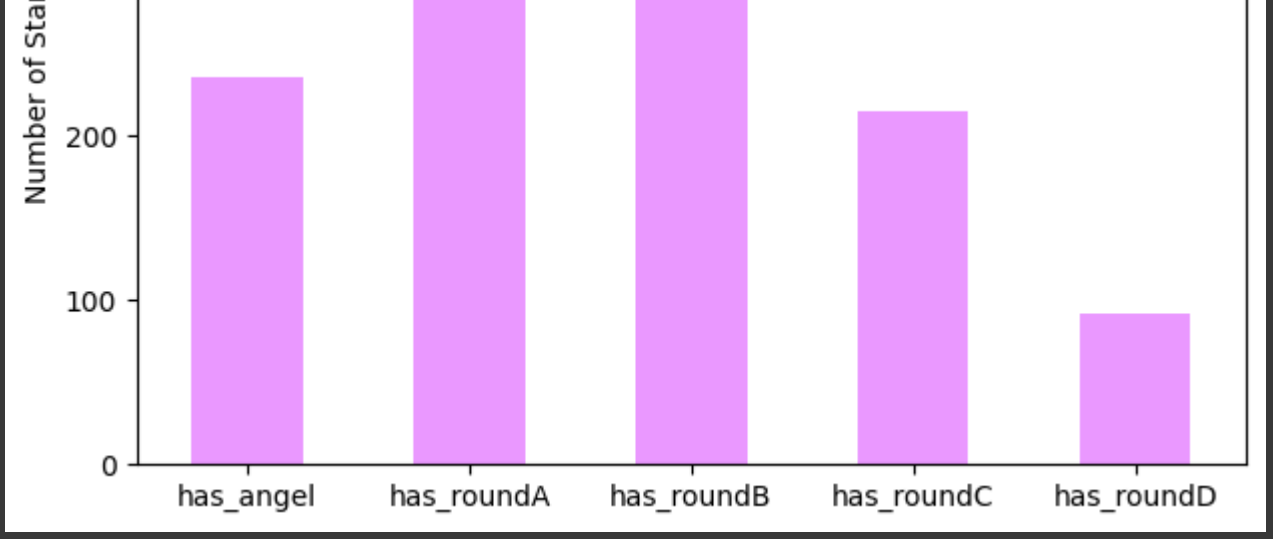


```
df.milestones.head(25).plot(kind='bar', color='pink')
#plt.xlabel('Number of Relationships')
plt.ylabel('Number of StartUps')
plt.title('Relationships')
plt.tight_layout()
plt.show()
```

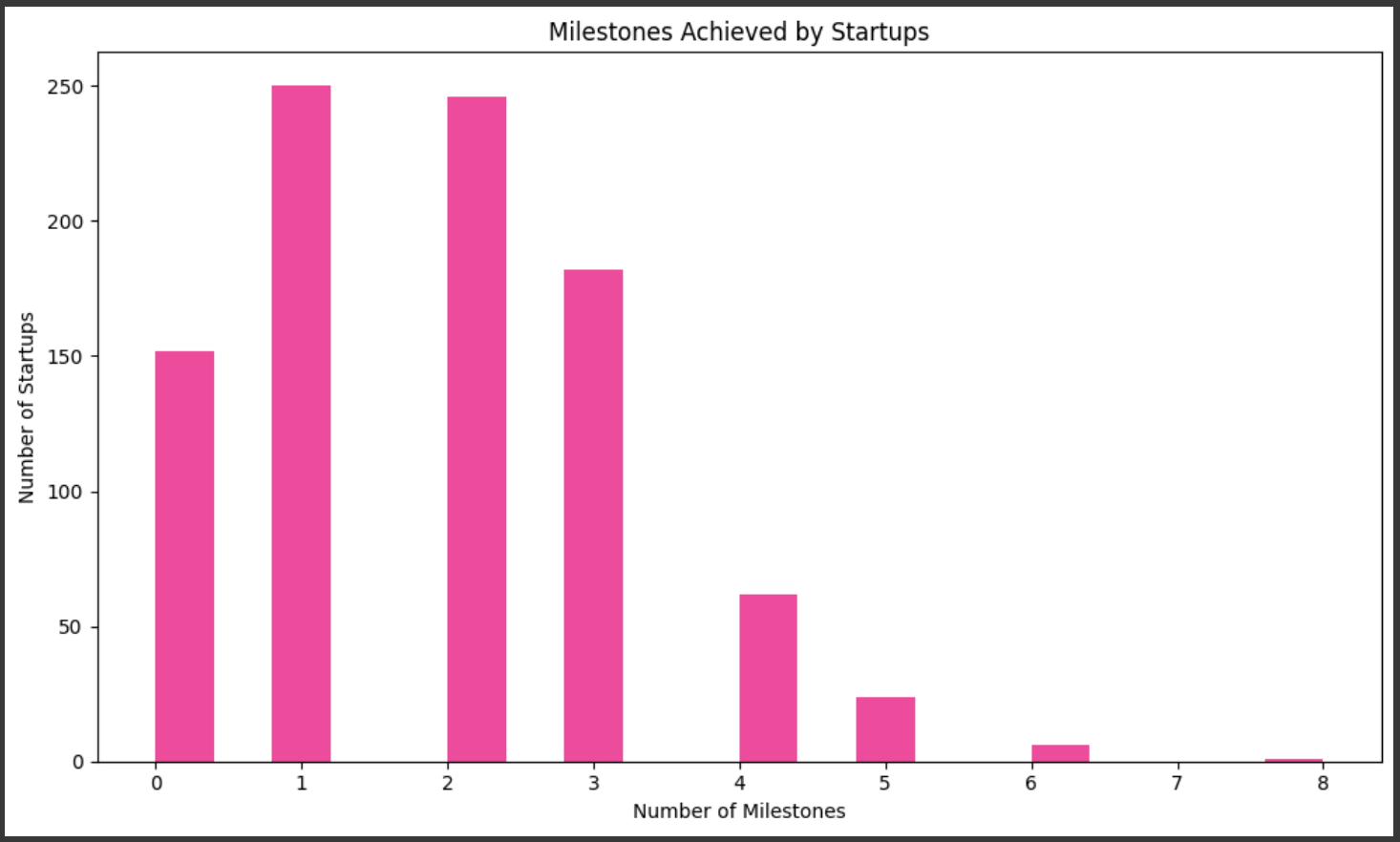



```
funding_rounds = df[['has_angel', 'has_roundA', 'has_roundB', 'has_roundC', 'has_roundD']].s


funding_rounds.plot(kind='bar', color='#d633ff', alpha=0.5)
#plt.xlabel('Funding Round')
plt.ylabel('Number of StartUps ')
plt.title('Number of StartUps with Different types of Funding')
plt.xticks(rotation=0)
plt.tight_layout()

# Show the plot
plt.show()
```





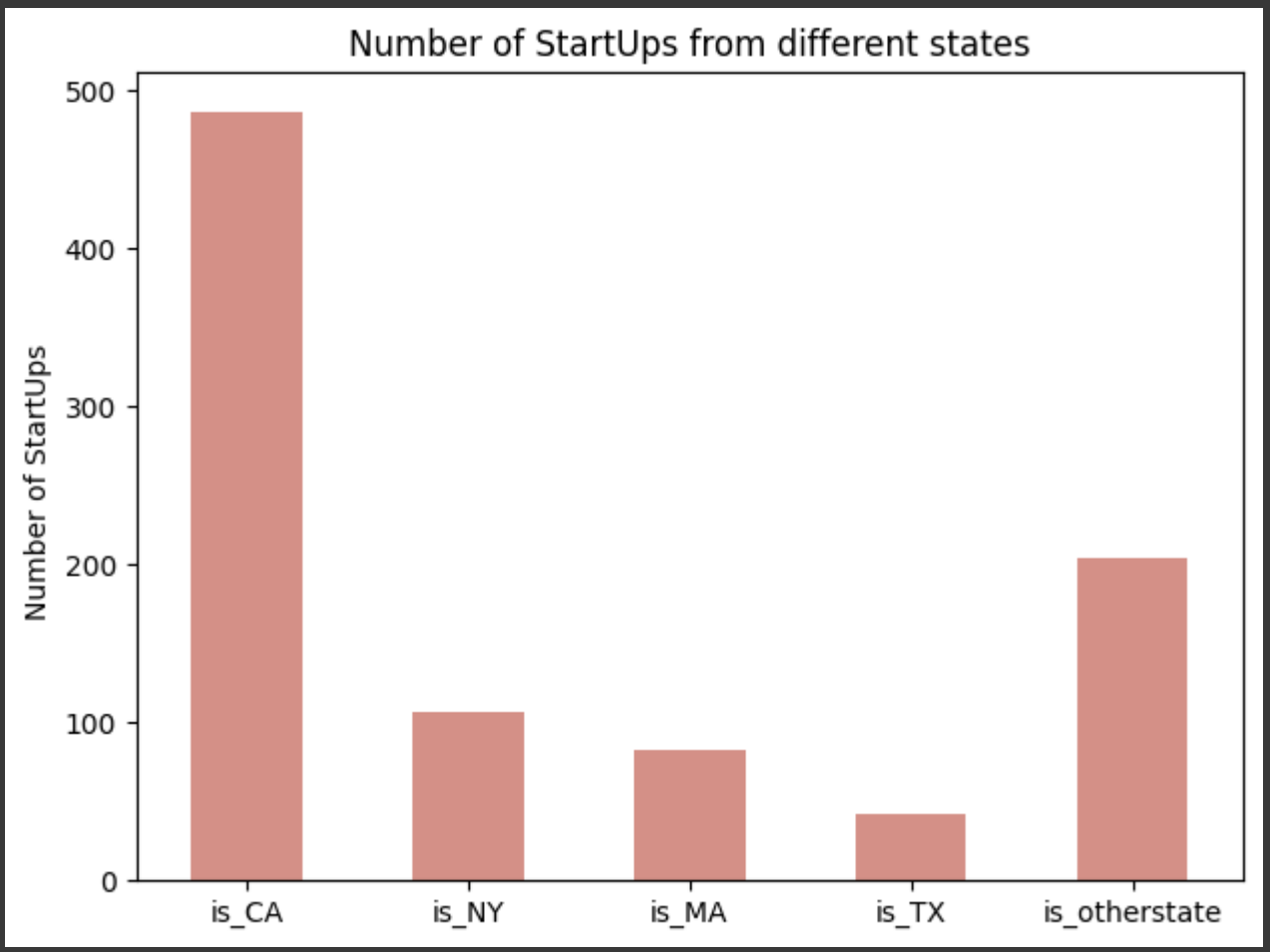

```
# Create a histogram of milestones
plt.figure(figsize=(10, 6))
plt.hist(df['milestones'], bins=20, color='#e60073', alpha=0.7)
plt.xlabel('Number of Milestones')
plt.ylabel('Number of Startups')
plt.title('Milestones Achieved by Startups')
plt.tight_layout()

# Show the chart
plt.show()
```



```
df.columns
```

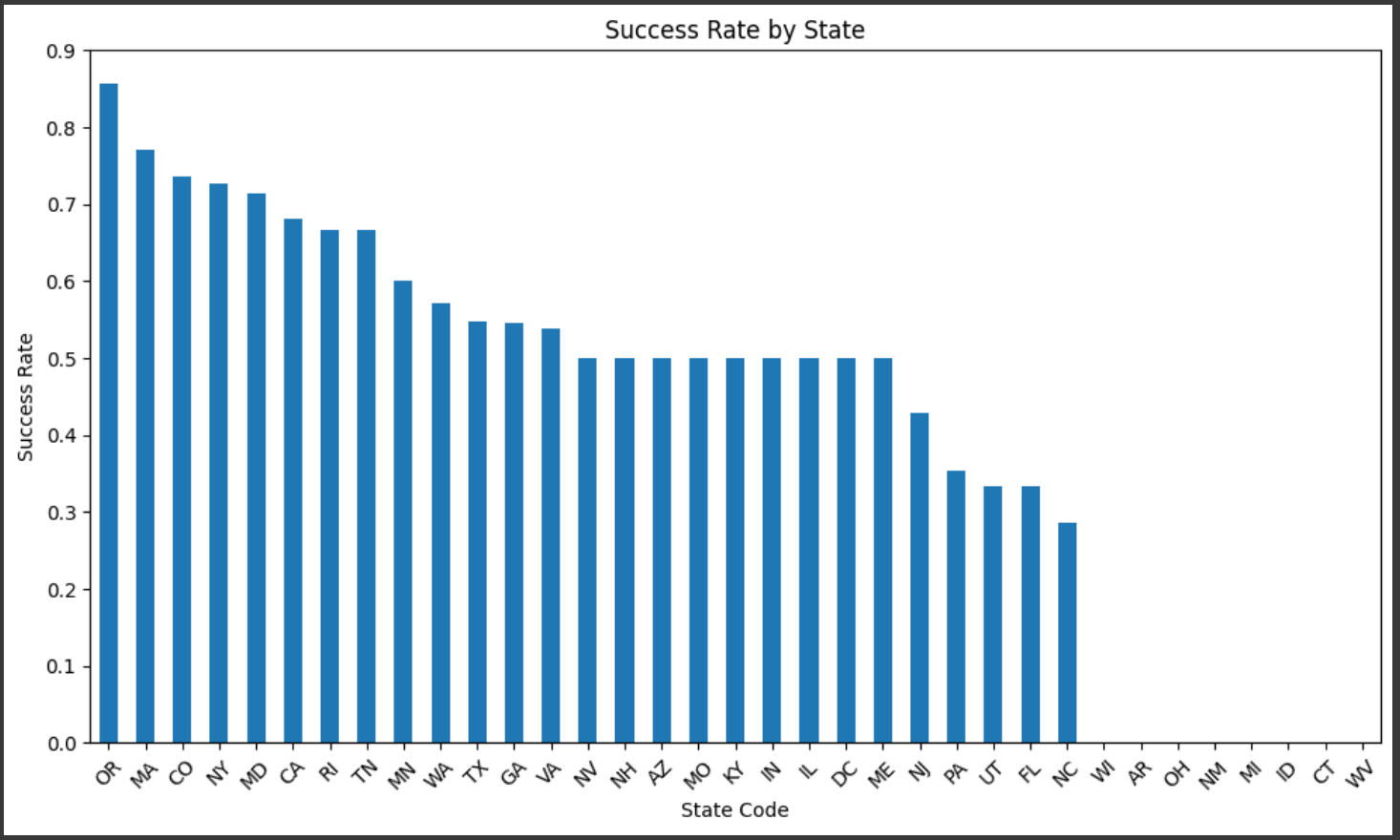
```
Index(['Unnamed: 0', 'state_code', 'latitude', 'longitude', 'zip_code', 'id',
       'city', 'Unnamed: 6', 'name', 'labels', 'founded_at', 'closed_at',
       'first_funding_at', 'last_funding_at', 'age_first_funding_year',
       'age_last_funding_year', 'age_first_milestone_year',
       'age_last_milestone_year', 'relationships', 'funding_rounds',
       'funding_total_usd', 'milestones', 'state_code.1', 'is_CA', 'is_NY',
       'is_MA', 'is_TX', 'is_otherstate', 'category_code', 'is_software',
       'is_web', 'is_mobile', 'is_enterprise', 'is_advertising',
       'is_gamesvideo', 'is_ecommerce', 'is_biotech', 'is_consulting',
       'is_othercategory', 'object_id', 'has_VC', 'has_angel', 'has_roundA',
       'has_roundB', 'has_roundC', 'has_roundD', 'avg_participants',
       'is_top500', 'status'],
      dtype='object')
```

```
features_state = df[['is_CA', 'is_NY','is_MA', 'is_TX', 'is_otherstate']].sum()

features_state.plot(kind='bar', color='#a210', alpha=0.5)
#plt.xlabel('Funding Round')
plt.ylabel('Number of StartUps ')
plt.title('Number of StartUps from different states')
plt.xticks(rotation=0)
plt.tight_layout()

# Show the plot
plt.show()
```




```
# Assuming df is your DataFrame
success_by_state = df.groupby('state_code')['status_encoded'].mean().sort_values(ascending=F

plt.figure(figsize=(10, 6))
```

```
success_by_state.plot(kind='bar')
plt.title('Success Rate by State')
plt.xlabel('State Code')
plt.ylabel('Success Rate')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



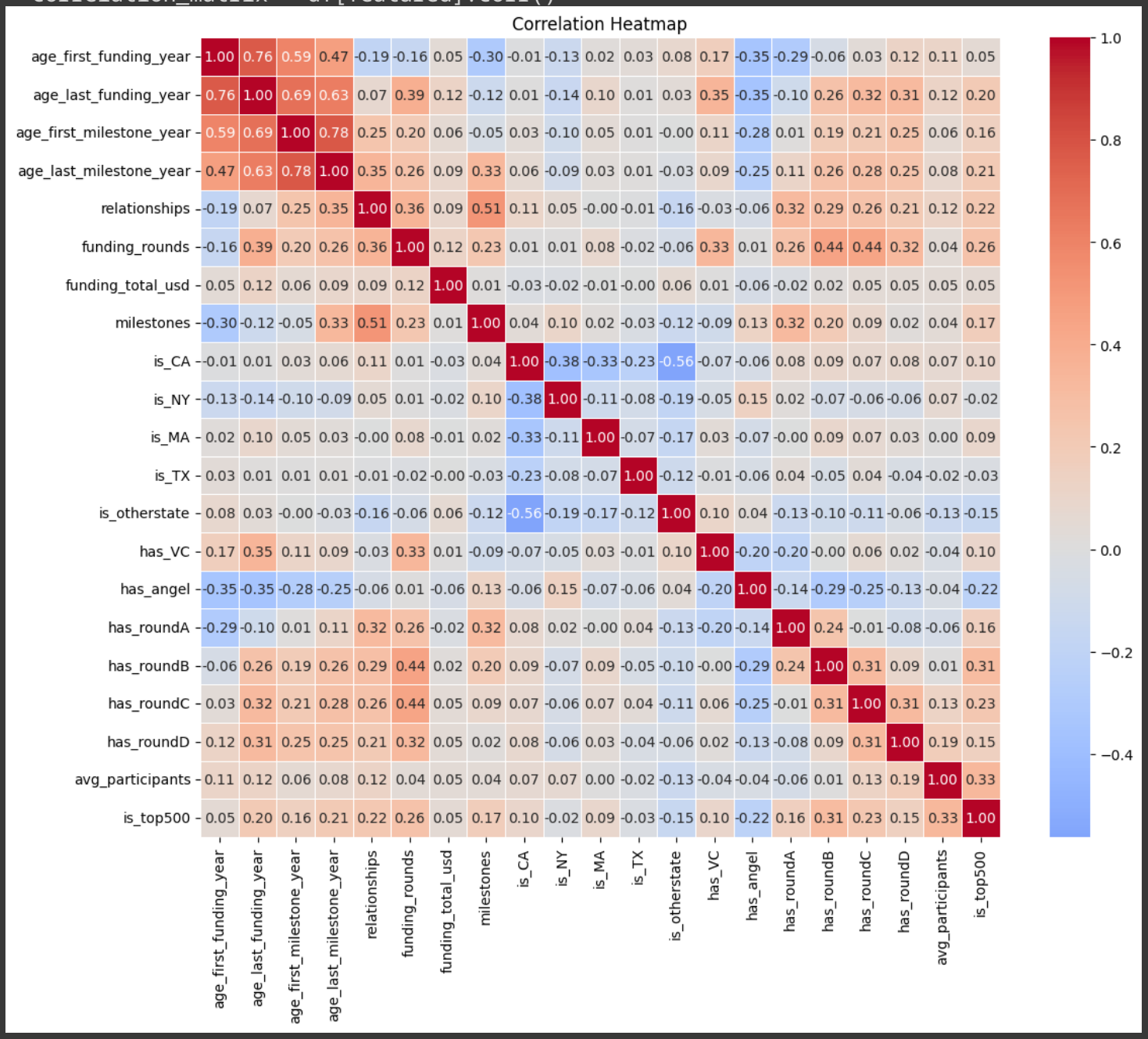

```
df.is_top500.value_counts()
```

```
1    747
0    176
Name: is_top500, dtype: int64
```

```
featured = ['founded_at', 'closed_at', 'first_funding_at',
       'last_funding_at', 'age_first_funding_year', 'age_last_funding_year',
       'age_first_milestone_year', 'age_last_milestone_year', 'relationships',
       'funding_rounds', 'funding_total_usd', 'milestones', 'is_CA', 'is_NY',
       'is_MA', 'is_TX', 'is_otherstate', 'has_VC', 'has_angel', 'has_roundA', 'has_roundB',
       'has_roundC', 'has_roundD', 'avg_participants', 'is_top500']

# Create a correlation matrix for the selected columns
correlation_matrix = df[featured].corr()

# Create a heatmap using Seaborn
plt.figure(figsize=(12, 10))
```

```
sb.heatmap(correlation_matrix, annot=True, cmap='coolwarm', center=0, square=True,  fmt=".2f
plt.title('Correlation Heatmap')
plt.tight_layout()


# Show the plot

plt.show()
```

```
<ipython-input-3-07c6001a2d3e>:9: FutureWarning: The default value of numeric_only in D
  correlation_matrix = df[featured].corr()
```



```
#from pandas_profiling import ProfileReport as pp
```

```
#profile = pp(df, title="this", html={'style':{'full_width':False}})
```

```
#profile.to_notebook_iframe()
```

```
#profile.to_file(output_file="reported.html")
```

The Real Deal

```
#importing ML libs
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

```
# features to use for prediction
features = ['age_first_funding_year', 'age_last_funding_year', 'funding_total_usd', 'is_CA',
            'is_TX', 'is_otherstate', 'is_software', 'is_web', 'is_mobile', 'is_enterprise',
            'is_ecommerce', 'is_biotech', 'is_consulting', 'is_othercategory', 'has_VC', 'ha
            'has_roundC', 'has_roundD', 'avg_participants', 'is_top500']

# Create a new column 'success' where 'Acquired' is 1 and 'Closed' is 0
df['success'] = df['status_encoded'].apply(lambda x: 1 if x == 1 else 0)

# Select features and target variable
X = df[features]
y = df['success']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create and train a Random Forest Classifier
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)

# Predict the success status on the test set
y_pred = model.predict(X_test)

# Generate a classification report
report = classification_report(y_test, y_pred)
print(report)
```

Printing with each Actual Results to compare with

```
#importing again if someone forgets to run the above
import pandas as pd
```

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report


# Select the features you want to use for prediction
features = ['age_first_funding_year', 'age_last_funding_year', 'funding_total_usd', 'is_CA',
            'is_otherstate', 'is_software', 'is_web', 'is_mobile', 'is_enterprise', 'is_adve
            'is_ecommerce', 'is_biotech', 'is_consulting', 'is_othercategory', 'has_VC', 'ha
            'has_roundB', 'has_roundC', 'has_roundD', 'avg_participants', 'is_top500']

# Create a new column 'success' where 'Acquired' is 1 and 'Closed' is 0
df['success'] = df['status_encoded'].apply(lambda x: 1 if x == 1 else 0)

# features and target variable
X = df[features]
y = df['success']

# Spliting data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Random Forest Classifier
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)

# Predict the success status on the test set
y_pred = model.predict(X_test)

# Print each test data point, predicted result, and actual result
for i in range(len(X_test)):
    print(f"Test Data: {X_test.iloc[i]}")
    print(f"Predicted Result: {'Acquired' if y_pred[i] == 1 else 'Closed'}")
    print(f"Actual Result: {'Acquired' if y_test.iloc[i] == 1 else 'Closed'}")
    print("=" * 30)

# Generate a classification report
report = classification_report(y_test, y_pred)
print(report)
```

Giving Random Data to test

```
new_data_success = {
    'age_first_funding_year': 2,
    'age_last_funding_year': 5,
    'funding_total_usd': 8000000,
    'is_CA': 1,
    'is_NY': 0,
    'is_MA': 0,
    'is_TX': 0,
    'is_otherstate': 0,
    'is_software': 0,
    'is_web': 0,
    'is_mobile': 0,
    'is_enterprise': 0,
    'is_advertising': 0,
```

```
    'is_gamesvideo': 0,
    'is_ecommerce': 0,
    'is_biotech': 0,
    'is_consulting': 0,
    'is_othercategory': 1,
    'has_VC': 1,
    'has_angel': 1,
    'has_roundA': 1,
    'has_roundB': 1,
    'has_roundC': 0,
    'has_roundD': 0,
    'avg_participants': 1,
    'is_top500': 0
}


# Convert the new data into a DataFrame
new_df = pd.DataFrame([new_data_success])

# Use the trained model to predict the outcome for the new data
predicted_result = model.predict(new_df)

# Interpret the prediction
if predicted_result[0] == 1:
    print("The startup is predicted to be Acquired.")
else:
    print("The startup is predicted to be Closed.")
```

```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# Select the features you want to use for prediction
features = ['age_first_funding_year', 'age_last_funding_year', 'funding_total_usd', 'is_CA',

# LOL we forgot to spell correctly
df['success'] = df['status_encoded'].apply(lambda x: 1 if x == 1 else 0)

# Select features and target variable
X = df[features]
y = df['success']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=52)

# Create and train a Decision Tree Classifier
model = DecisionTreeClassifier(random_state=42)
model.fit(X_train, y_train)

# Predict the success status on the test set
y_pred = model.predict(X_test)

# Generate a classification report
report = classification_report(y_test, y_pred)
print(report)
```

Colab paid products - Cancel contracts here