Sigmoid Exam

Data Analysis

Raw Data:

company	rank	rank_ch	revenue	profit	num. of emplo	sector	city	state	newco	ceo_fou	ceo_w	profitat	prev_	CEO	Website	Ticker	Market Cap
Walmart	1	0	572754	13673	2300000	Retailing	Bentonville	AR	no	no	no	yes	1	C. Douglas McMillon	https://www.stock.walmart.com	WMT	352037
Amazon	2	0	469822	33364	1608000	Retailing	Seattle	WA	no	no	no	yes	2	Andrew R. Jassy	www.amazon.com	AMZN	1202717
Apple	3	0	365817	94680	154000	Technology	Cupertino	CA	no	no	no	yes	3	Timothy D. Cook	www.apple.com	AAPL	2443962
CVS Health	4	0	292111	7910	258000	Health Care	Woonsocket	RI	no	no	yes	yes	4	Karen Lynch	https://www.cvshealth.com	CVS	125204
UnitedHealth Group	5	0	287597	17285	350000	Health Care	Minnetonka	MN	no	no	no	yes	5	Andrew P. Witty	www.unitedhealthgroup.com	UNH	500468
Exxon Mobil	6	4	285640	23040	63000	Energy	Irving	TX	no	no	no	yes	10	Darren W. Woods	www.exxonmobil.com	XOM	371841
Berkshire Hathaway	7	-1	276094	89795	372000	Financials	Omaha	NE	no	no	no	yes	6	Warren E. Buffett	www.berkshirehathaway.com	BRKA	625468
Alphabet	8	1	257637	76033	156500	Technology	Mountain View	CA	no	no	no	yes	9	Sundar Pichai	https://www.abc.xyz	GOOGL	1309359
McKesson	9	-2	238228	-4539	67500	Health Care	Irving	TX	no	no	no	no	7	Brian S. Tyler	www.mckesson.com	MCK	47377
AmerisourceBergen	10	-2	213988.8	1539.9	40000	Health Care	Conshohocken	PA	no	no	no	yes	8	Steven H. Collis	www.amerisourcebergen.com	ABC	29972
Costco Wholesale	11	1	195929	5007	288000	Retailing	Issaquah	WA	no	no	no	yes	12	W. Craig Jelinek	www.costco.com	COST	230443
Cigna	12	1	174078	5365	72963	Health Care	Bloomfield	CT	no	no	no	yes	13	David Cordani	https://www.cigna.com	CI	88459
AT&T	13	-2	168864	20081	202600	Telecommunications	Dallas	TX	no	no	no	yes	11	John T. Stankey	www.att.com	T	148907
Microsoft	14	1	168088	61271	181000	Technology	Redmond	WA	no	no	no	yes	15	Satya Nadella	www.microsoft.com	MSFT	1941033
Cardinal Health	15	-1	162467	611	46827	Health Care	Dublin	OH	no	no	no	yes	14	Jason Hollar	www.cardinalhealth.com	CAH	15169
Chevron	16	11	162465	15625	42595	Energy	San Ramon	CA	no	no	no	yes	27	Michael K. Wirth	www.chevron.com	CVX	284132
Home Depot	17	1	151157	16433	490600	Retailing	Atlanta	GA	no	no	no	yes	18	Edward P. Decker	www.homedepot.com	HD	308152
Walgreens Boots Alliance	18	-2	148579	2542	258500	Food & Drug Stores	Deerfield	IL	no	no	yes	yes	16	Roz Brewer	www.walgreensbootsalliance.com	WBA	33360
Marathon Petroleum	19	13	141032	9738	17700	Energy	Findlay	OH	no	no	no	yes	32	Michael J. Hennigan	www.marathonpetroleum.com	MPC	47526
Elevance Health	20	3	138639	6104	98200	Health Care	Indianapolis	IN	no	no	yes	yes	23	Gail K. Boudreaux	www.elevancehealth.com	ELV	119923
Kroger	21	-4	137888	1655	420000	Food & Drug Stores	Cincinnati	OH	no	no	no	yes	17	W. Rodney McMullen	www.thekrogerco.com	KR	33846
Ford Motor	22	-1	136341	17937	183000	Motor Vehicles & Parts	Dearborn	MI	no	no	no	yes	21	James D. Farley Jr.	www.ford.com	F	50609
Verizon Communications	23	-3	133613	22065	118400	Telecommunications	New York	NY	no	no	no	yes	20	Hans E. Vestberg	www.verizon.com	VZ	211872
JPMorgan Chase	24	-5	127202	48334	271025	Financials	New York	NY	no	no	no	yes	19	James Dimon	www.jpmorganchase.com	JPM	336469
General Motors	25	-3	127004	10019	157000	Motor Vehicles & Parts	Detroit	MI	no	no	yes	yes	22	Mary T. Barra	www.gm.com	GM	50156
Centene	26	-2	125982	1347	72500	Health Care	St. Louis	MO	no	no	yes	yes	24	Sarah M. London	www.centene.com	CNC	53429
Meta Platforms	27	7	117929	39370	71970	Technology	Menlo Park	CA	no	yes	no	yes	34	Mark Zuckerberg	https://investor.fb.com	META	475718
Comcast	28	-2	116385	14159	189000	Telecommunications	Philadelphia	PA	no	no	no	yes	26	Brian L. Roberts	www.comcastcorporation.com	CMCSA	185069
Phillips 66	29	19	114852	1317	14000	Energy	Houston	TX	no	no	no	yes	48	Mark E. Lashier	www.phillips66.com	PSX	41091
Valero Energy	30	23	108332	930	9804	Energy	San Antonio	TX	no	no	no	yes	53	Joseph W. Gorder	www.valero.com	VLO	44376

General info

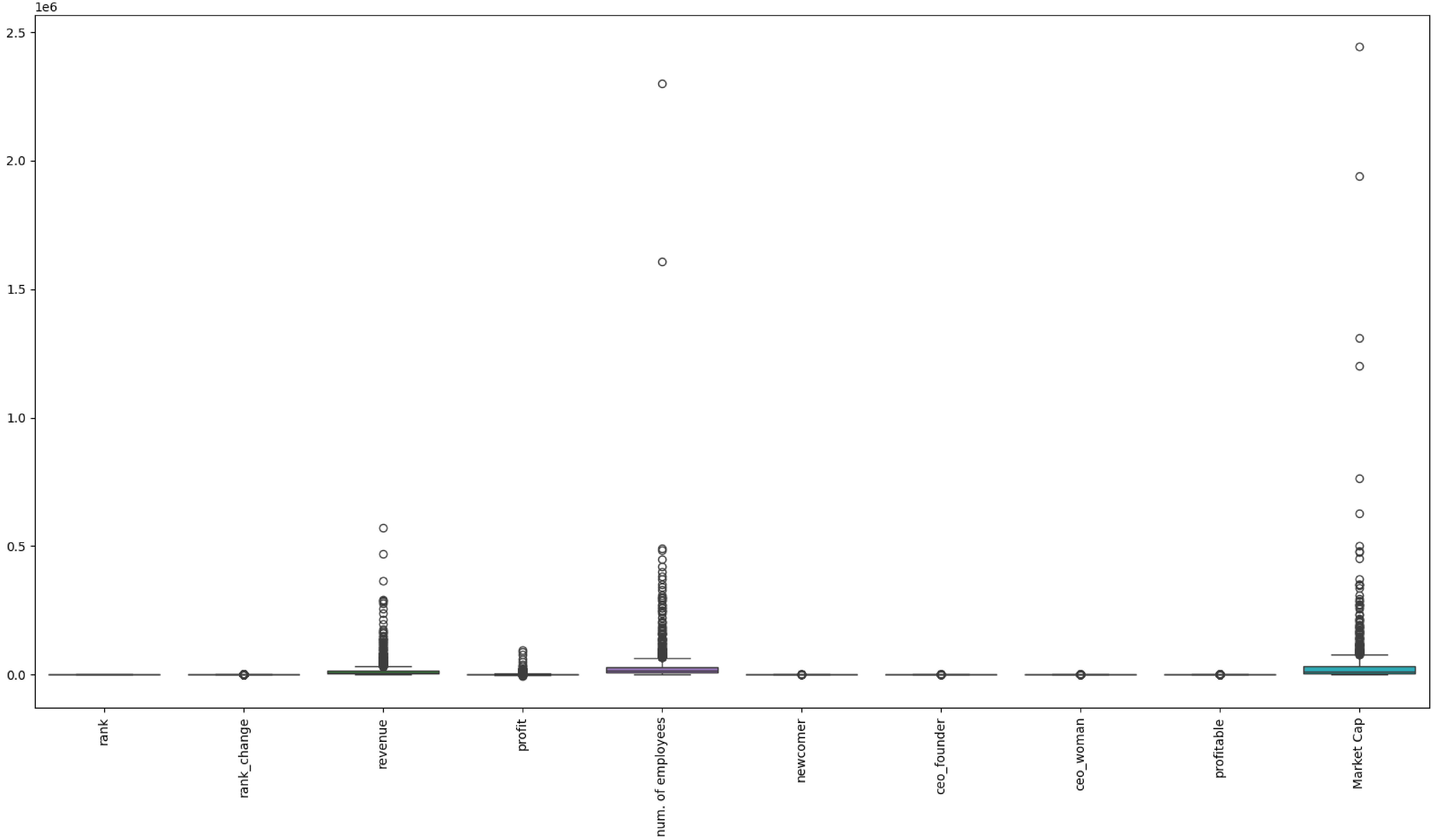
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 18 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   company            1000 non-null   object
 1   rank               1000 non-null   int64
 2   rank_change        1000 non-null   float64
 3   revenue            1000 non-null   float64
 4   profit             997 non-null    float64
 5   num. of employees  999 non-null    float64
 6   sector             1000 non-null   object
 7   city               1000 non-null   object
 8   state              1000 non-null   object
 9   newcomer           1000 non-null   object
 10  ceo_founder        1000 non-null   object
 11  ceo_woman          1000 non-null   object
 12  profitable         1000 non-null   object
 13  prev_rank          1000 non-null   object
 14  CEO                1000 non-null   object
 15  Website            1000 non-null   object
 16  Ticker             951 non-null    object
 17  Market Cap         969 non-null    object
dtypes: float64(4), int64(1), object(13)
memory usage: 140.8+ KB
```

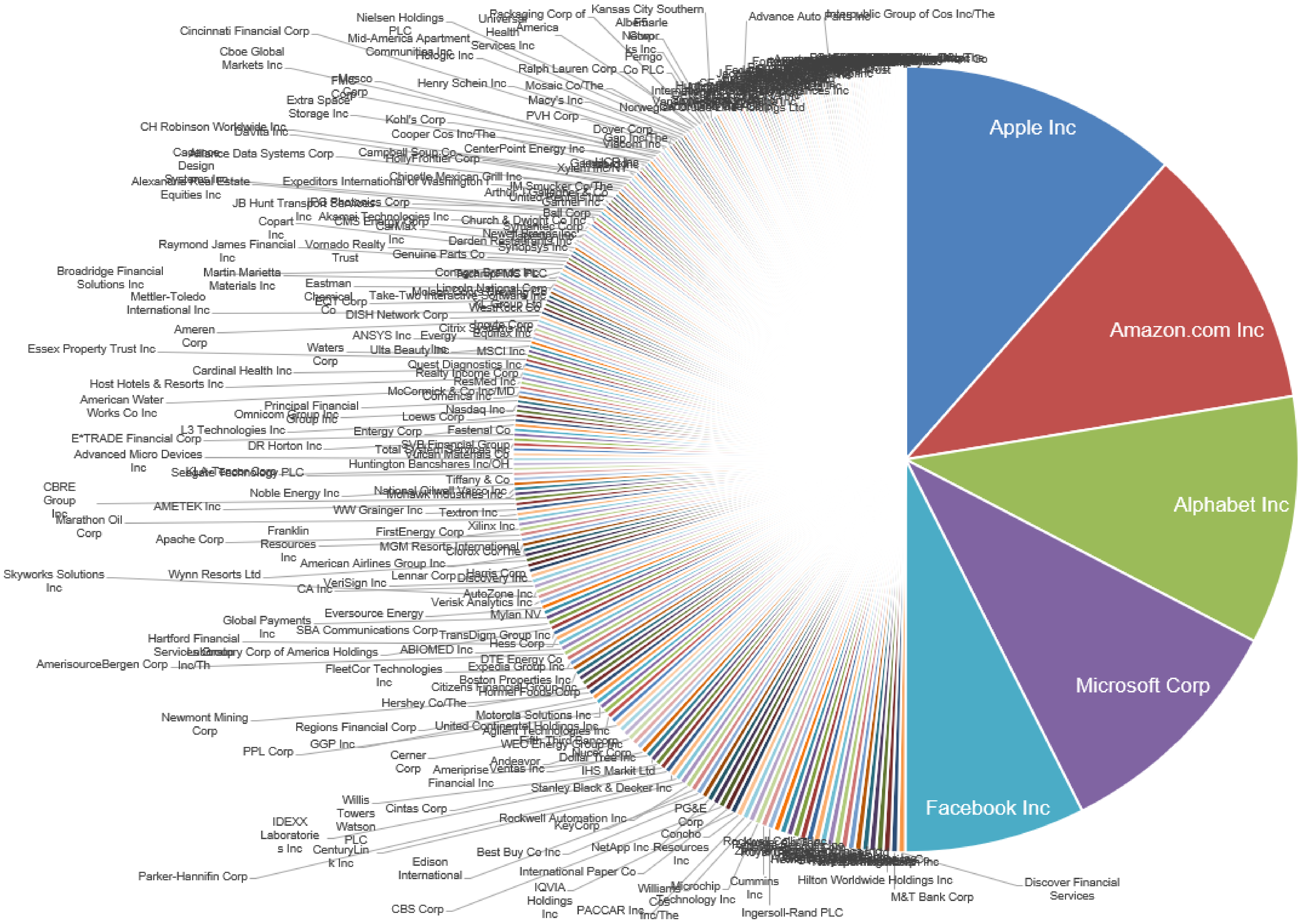
```
numerical_features = df[['revenue', 'profit', 'num. of employees', 'Market Cap']
numerical_features_description_updated = numerical_features.describe()
```

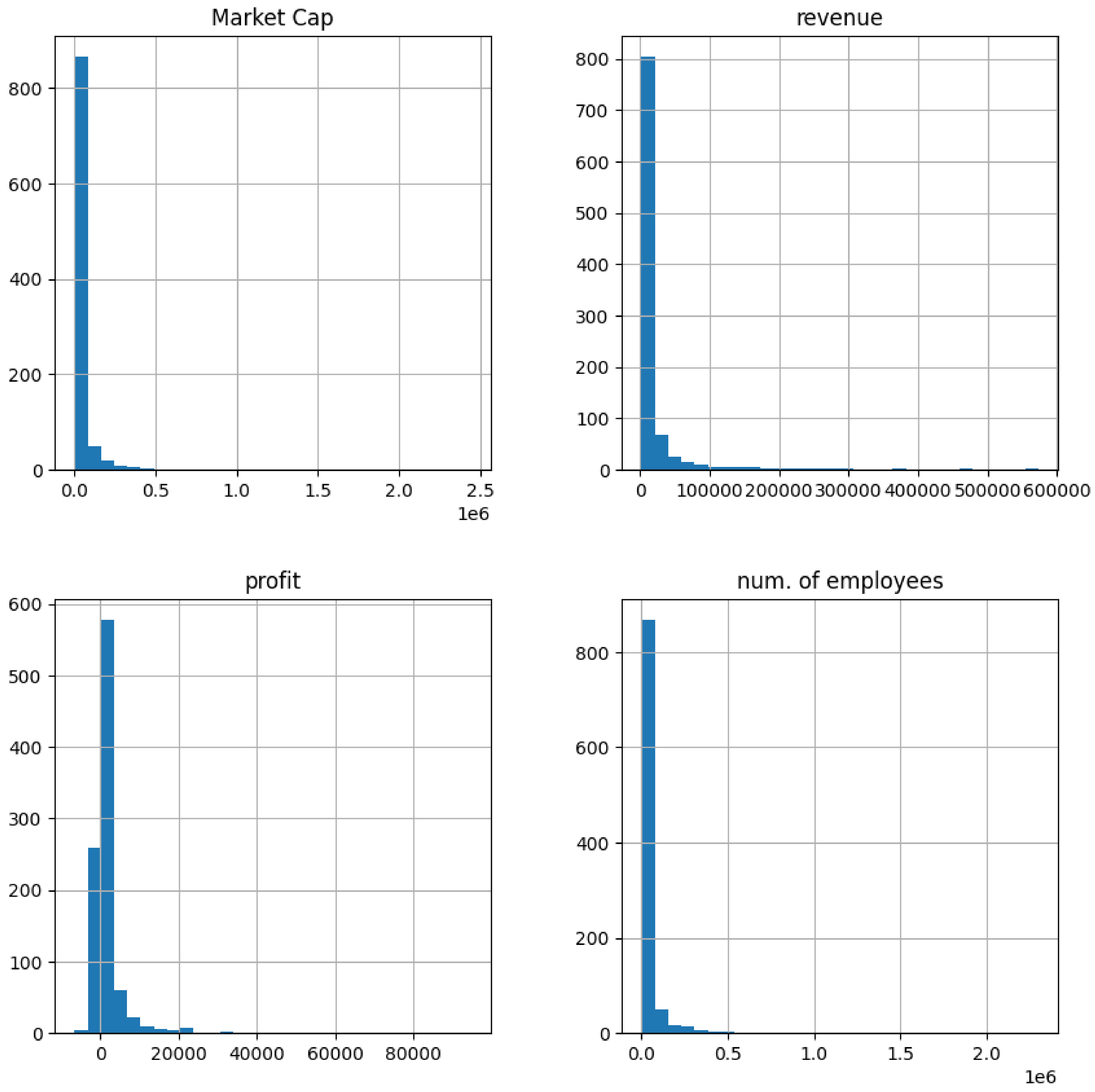
	revenue	profit	num. of employees	Market Cap
count	961.000000	958.000000	9.610000e+02	9.610000e+02
mean	17999.396462	2070.201983	3.651696e+04	4.002294e+04
std	41463.958721	6543.255875	1.063753e+05	1.309925e+05
min	2107.200000	-6520.000000	1.600000e+02	1.490000e+02
25%	3494.800000	193.500000	6.640000e+03	4.415600e+03
50%	6310.200000	573.150000	1.400000e+04	1.164160e+04
75%	14298.000000	1502.250000	3.000000e+04	3.345560e+04
max	572754.000000	94680.000000	2.300000e+06	2.443962e+06

Detecting outliers with boxplot



Distribution of Market Cap in S&P500

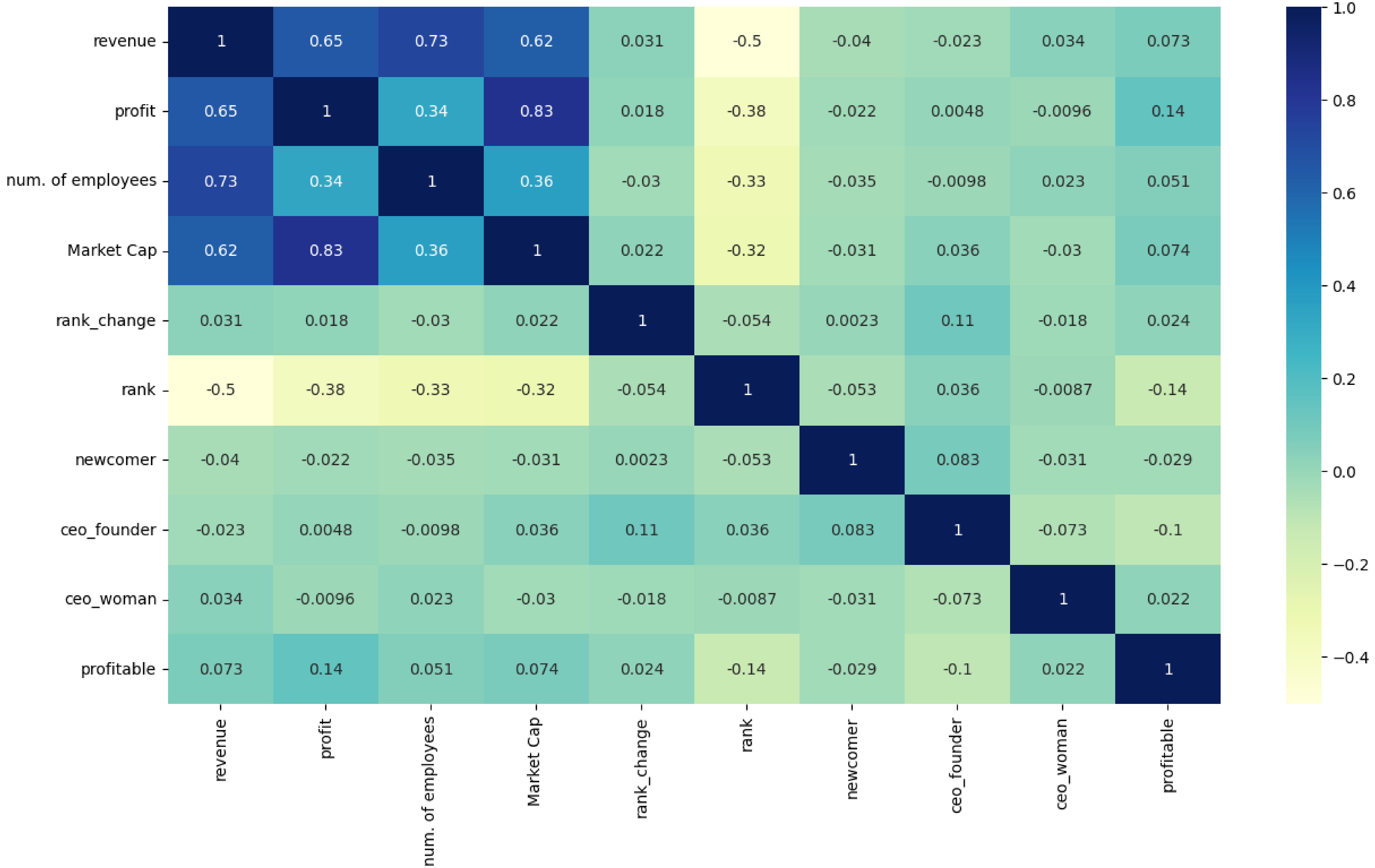




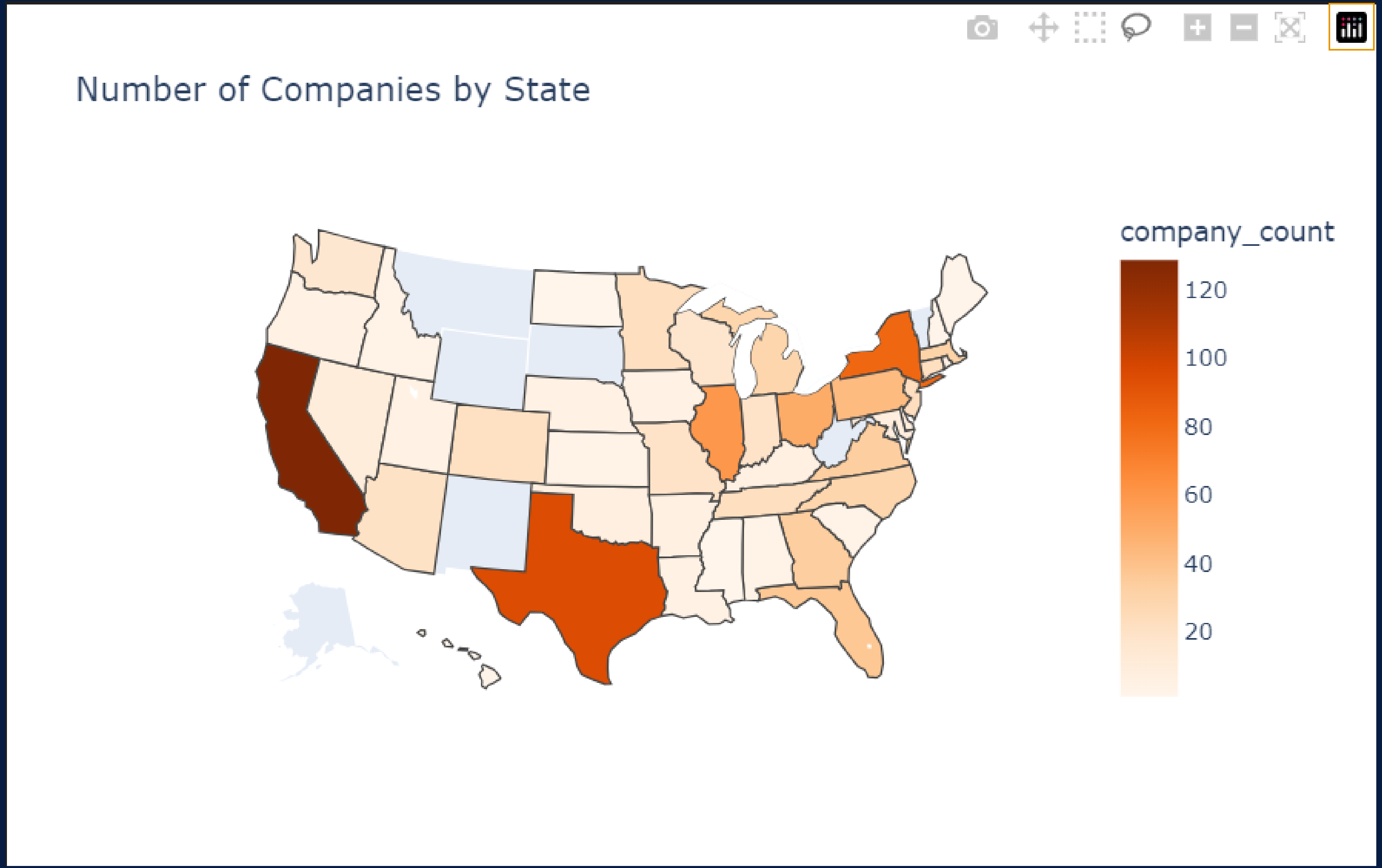
Histograms

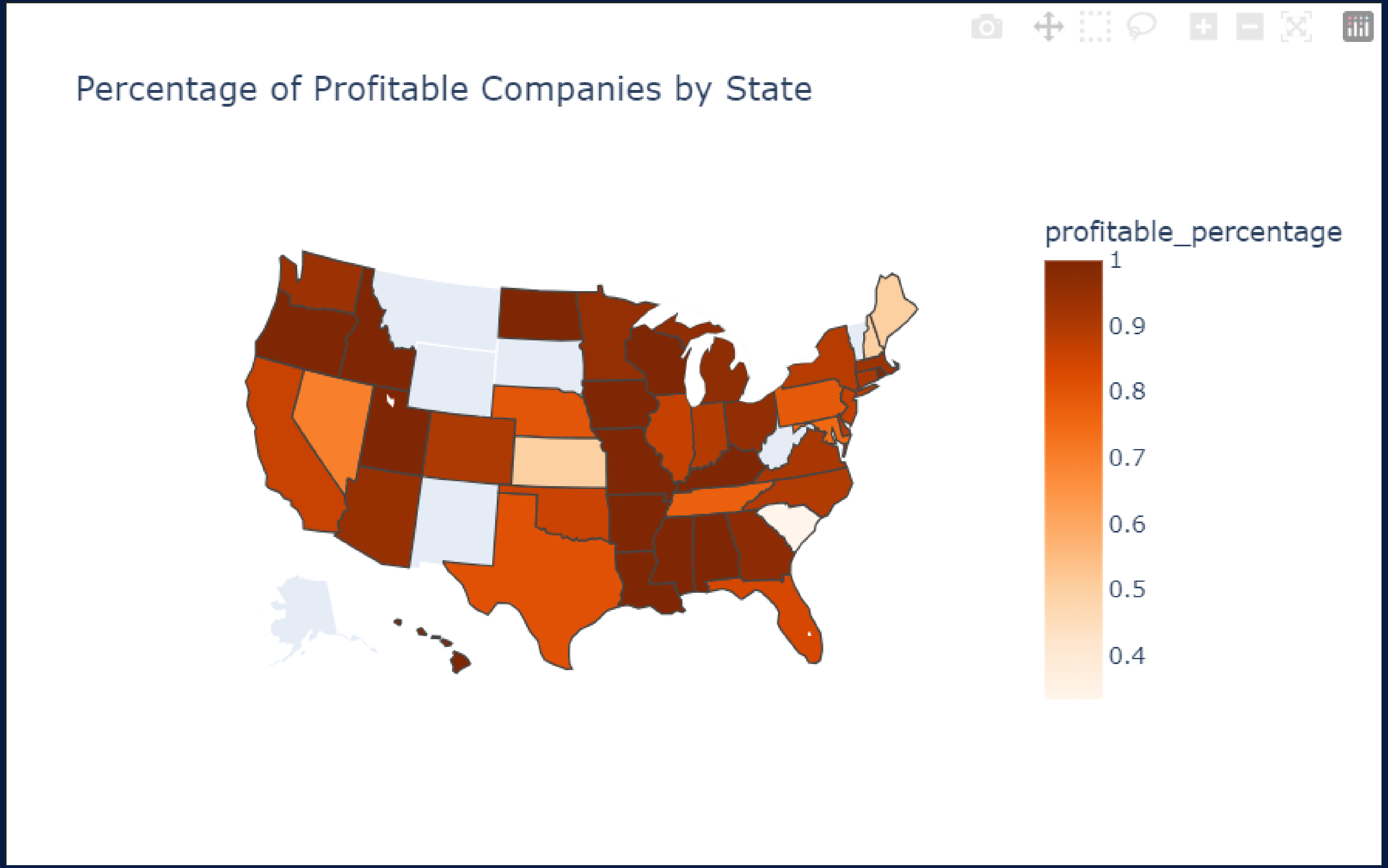
As we can see the majority of data is located close to 0, having small "Market Cap", "revenue", "num. of employees" and "profit". This means it does not have a Gaussian distribution and i should normalize it.

Correlation matrix

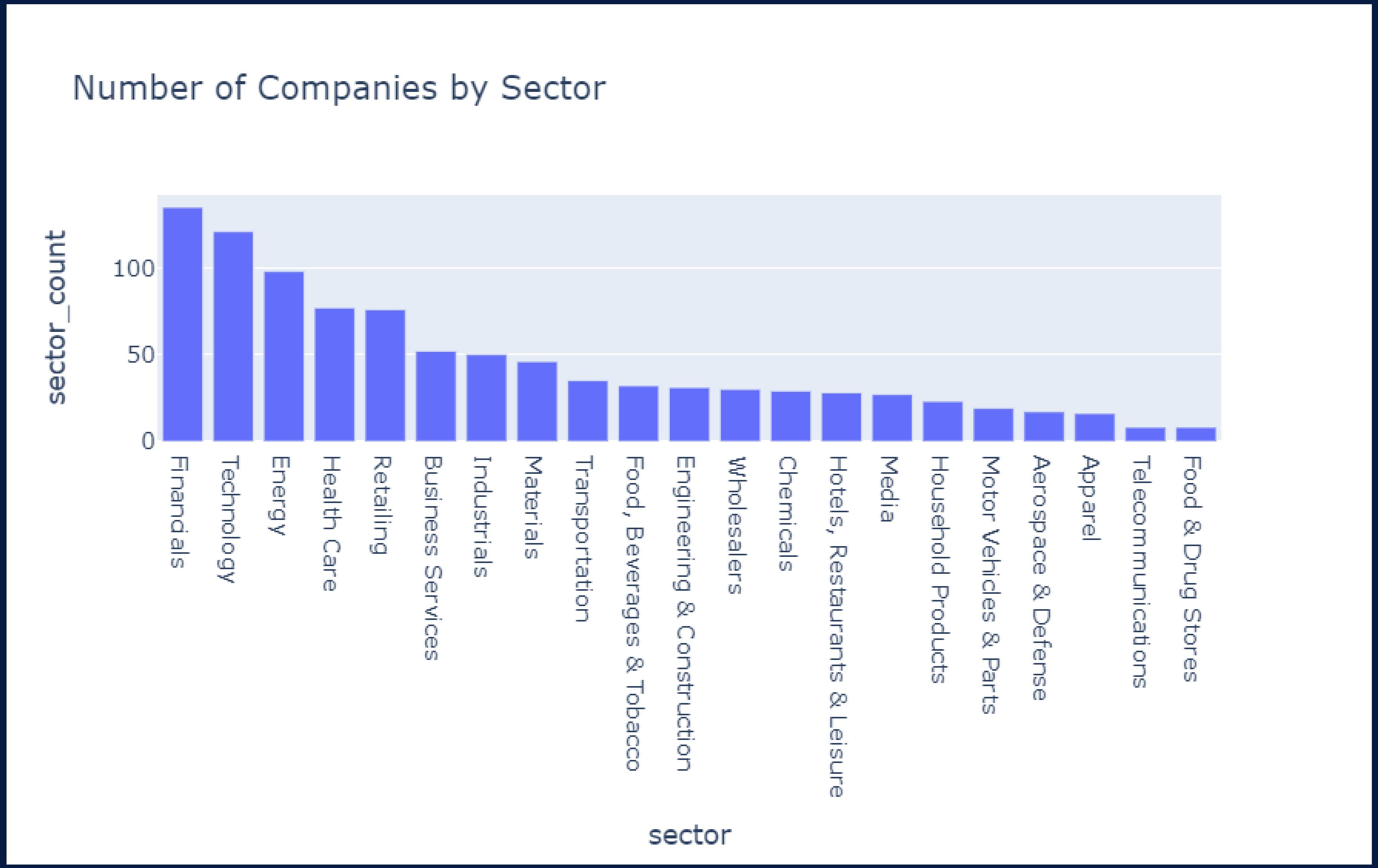


Looking trough cities and states





Exploring sector column



Data Preprocessing

1) Dropping unique columns for each company

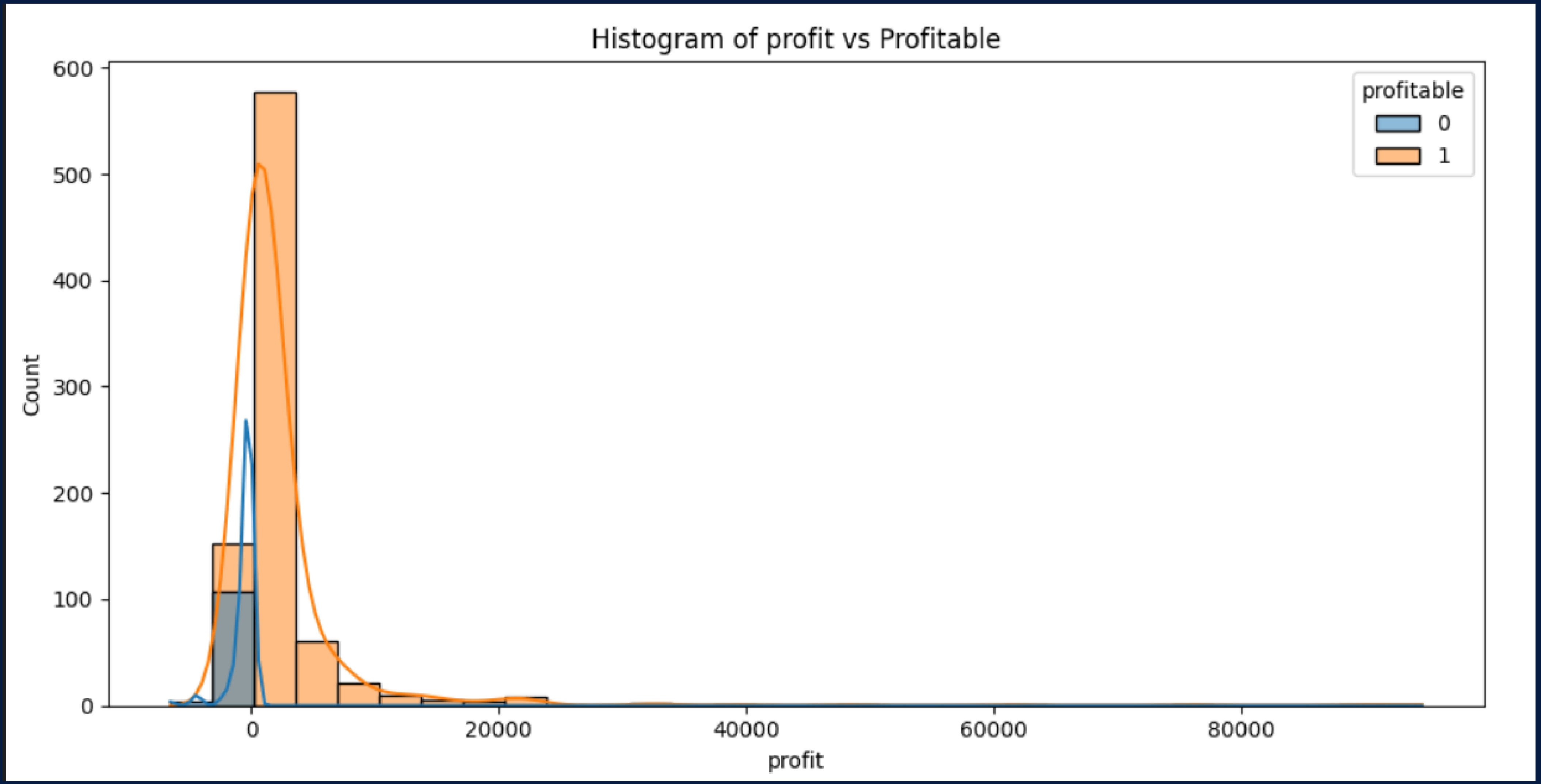
2) Dropping "prev_rank" as half values are missing

```
count_missing = df['prev_rank'].replace(' ', np.nan).astype(float).isna().sum()
print(f"Missing values in 'prev_rank' column: {count_missing}")
```

```
Missing values in 'prev_rank' column: 531
```

```
df = df.drop(['company', 'CEO', 'Website', 'Ticker'
              ,'prev_rank',
              'profit'], axis=1)
```

3) "profit" column as it directly affects target column



Is profit predicting profitable column ?

if profit <= 0:
profitable = 0
else:
profitable = 1

```
# Non-Positive profit and profitable companies
negative_companies = df[df['profit'] <= 0]
negative_profit = negative_companies['profitable'].value_counts()
negative_profit
```

```
profitable
0    111
```

Binary encoding + Missing values

```
columns_to_map = ['newcomer', 'ceo_founder',
                  'ceo_woman', 'profitable']
for column in columns_to_map:
    df[column] = df[column].map({'yes': 1, 'no': 0})
```

```
rank                   0
rank_change            0
revenue                0
num. of employees      1
sector                 0
city                   0
state                  0
newcomer               0
ceo_founder            0
ceo_woman              0
profitable             0
Market Cap            39
```

```
# Missing value imputation with KNNImputer

imputer = KNNImputer(n_neighbors=5)
# Getting the columns with missing values
columns_with_missing_values = df_1.columns[df_1.isna().any()].tolist()

for column in columns_with_missing_values:
    missing_column = df_1[column]
    missing_column_2d = missing_column.values.reshape(-1, 1)
    imputed_column_2d = imputer.fit_transform(missing_column_2d)
    df_1[column] = imputed_column_2d.flatten()



# Assigning in df_1 the non-numerical columns from df by rank
df_1 = pd.merge(df_1, concatting_columns_df, on='rank', how='left')
df_1 = df_1.drop('rank', axis=1)
df = df_1
df.info()
```

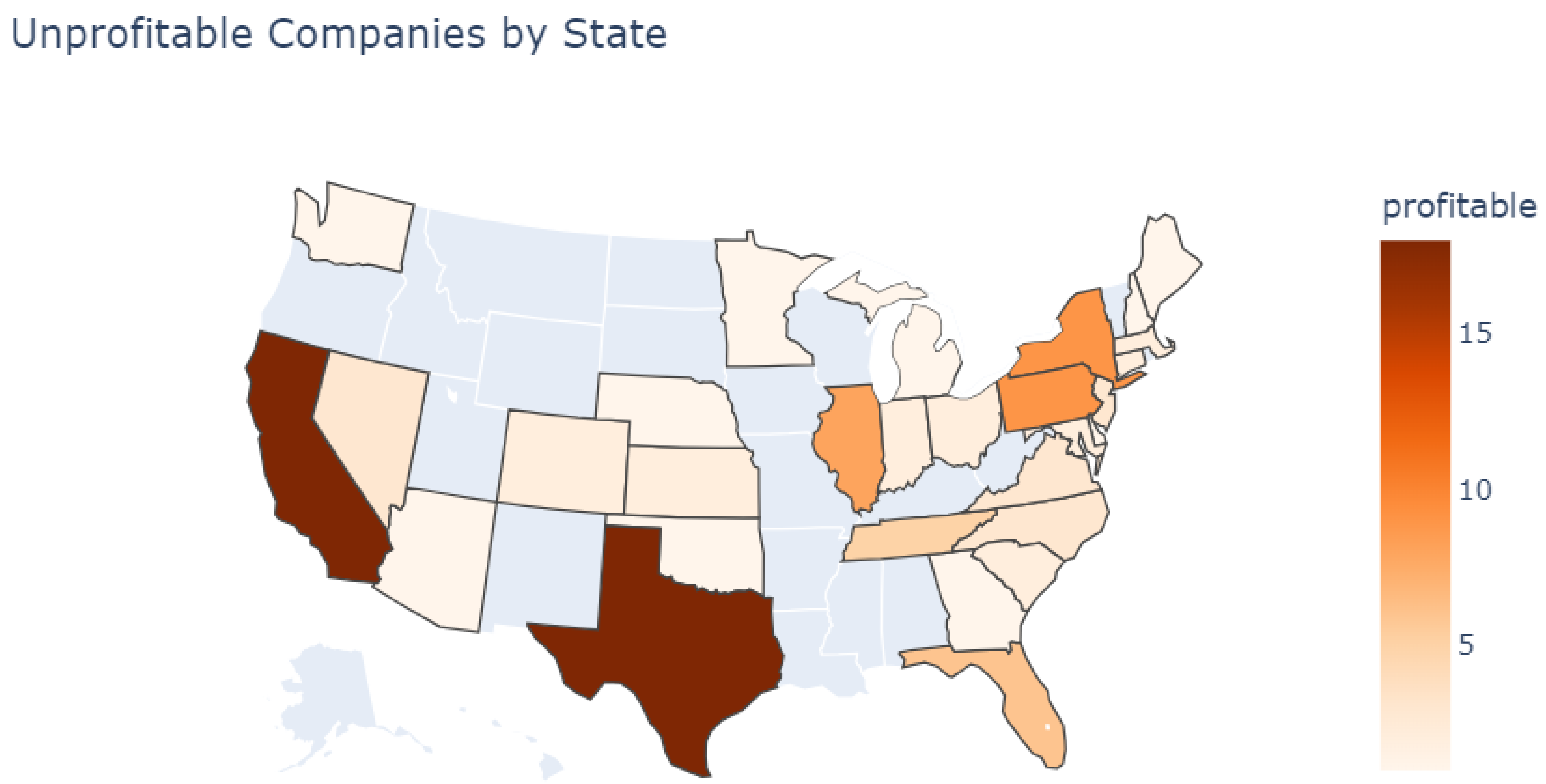
One Hot ‘city’ and ‘state’ columns

```
Data columns (total 11 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   revenue             1000 non-null   float64
 1   num. of employees   1000 non-null   float64
 2   newcomer            1000 non-null   int64
 3   ceo_founder         1000 non-null   int64
 4   ceo_woman           1000 non-null   int64
 5   profitable          1000 non-null   int64
 6   Market Cap          1000 non-null   float64
 7   sector              1000 non-null   object
 8   city                1000 non-null   object
 9   state               1000 non-null   object
 10  rank_change         1000 non-null   float64
```

```
 There are 400 cities
 There are 46 states
Top states by city count:

state
CA    50
PA    26
TX    22
IL    22
MI    22
OH    22
NJ    19
FL    19
MA    16
NY    16
```

```
Data columns (total 74 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   revenue             1000 non-null   float64
 1   num. of employees   1000 non-null   float64
 2   newcomer            1000 non-null   int64
 3   ceo_founder         1000 non-null   int64
 4   ceo_woman           1000 non-null   int64
 5   profitable          1000 non-null   int64
 6   Market Cap          1000 non-null   float64
 7   sector              1000 non-null   object
 8   rank_change         1000 non-null   float64
 9   state_CT            1000 non-null   bool
 10  state_FL            1000 non-null   bool
 11  state_GA            1000 non-null   bool
 12  state_IL            1000 non-null   bool
 13  state_MA            1000 non-null   bool
 14  state_MI            1000 non-null   bool
 15  state_MN            1000 non-null   bool
 16  state_NC            1000 non-null   bool
 17  state_NJ            1000 non-null   bool
 18  state_NY            1000 non-null   bool
 19  state_OH            1000 non-null   bool
...
 72  city_Wilmington     1000 non-null   bool
 73  city_city_other     1000 non-null   bool
```

One Hot ‘city’ and ‘state’ columns

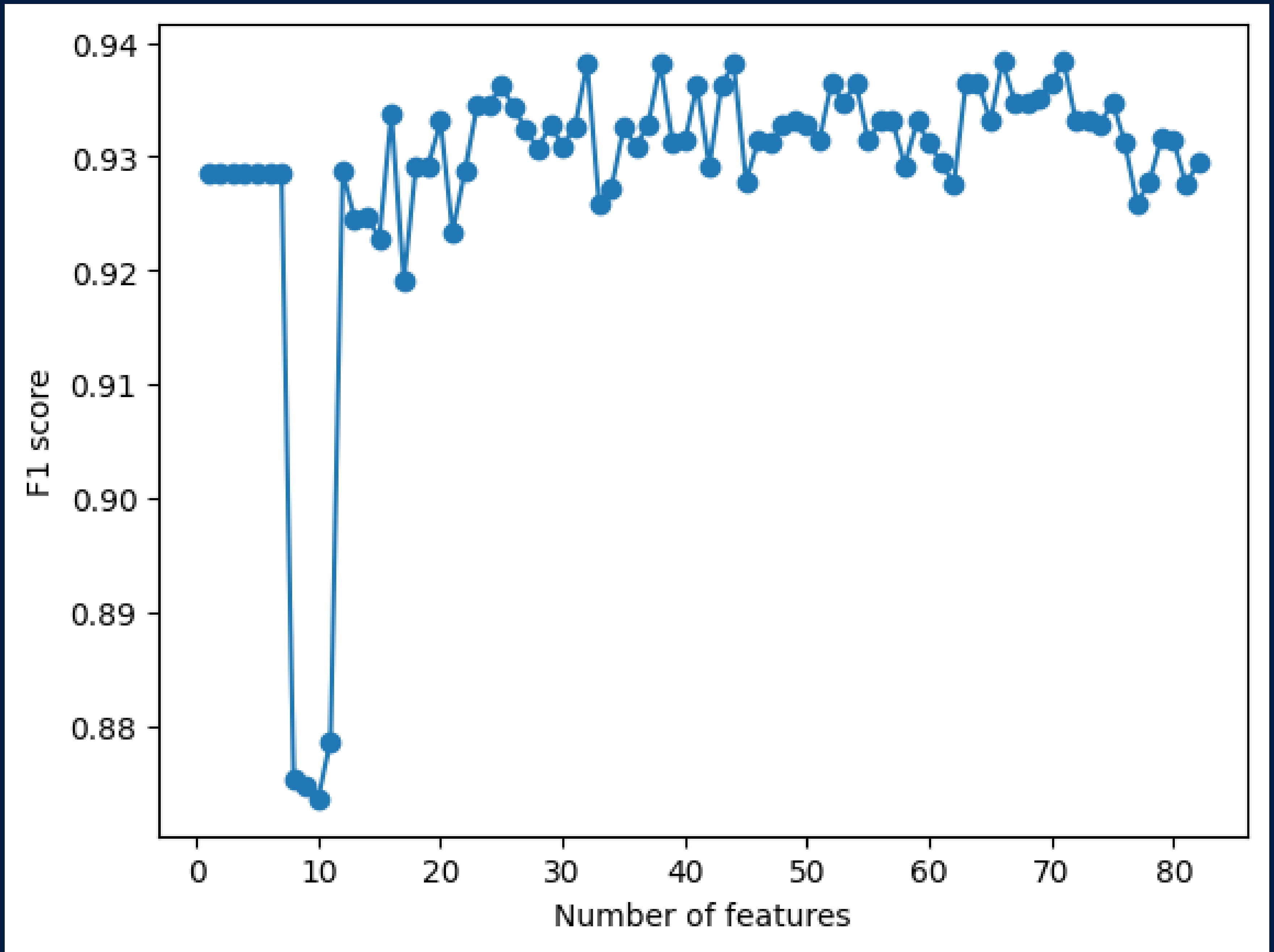


Same approach for sector column

```python
# One hot sector by top 10 and others
top_10 = df['sector'].value_counts().index[:10]
df['sector'] = df['sector'].apply(lambda x: x if x in top_10 else 'others')
df = pd.get_dummies(df, columns=['sector'], drop_first=True)
df.info()
```

```
Data columns (total 83 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   revenue                 1000 non-null   float64
 1   num. of employees       1000 non-null   float64
 2   newcomer                1000 non-null   int64
 3   ceo_founder             1000 non-null   int64
 4   ceo_woman               1000 non-null   int64
 5   profitable              1000 non-null   int64
 6   Market Cap              1000 non-null   float64
 7   rank_change             1000 non-null   float64
 8   state_CT                1000 non-null   bool
 9   state_FL                1000 non-null   bool
 10  state_GA                1000 non-null   bool
 11  state_IL                1000 non-null   bool
 12  state_MA                1000 non-null   bool
 13  state_MI                1000 non-null   bool
 14  state_MN                1000 non-null   bool
 15  state_NC                1000 non-null   bool
 16  state_NJ                1000 non-null   bool
 17  state_NY                1000 non-null   bool
 18  state_OH                1000 non-null   bool
 19  state_PA                1000 non-null   bool
...
 81  sector_Transportation   1000 non-null   bool
 82  sector_others           1000 non-null   bool
```

Choosing best number of features




```
Data columns (total 67 columns):
 #   Column                              Non-Null Count  Dtype
---  ------                              --------------  -----
 0   revenue                             1000 non-null   float64
 1   num. of employees                   1000 non-null   float64
 2   newcomer                            1000 non-null   int64
 3   ceo_founder                         1000 non-null   int64
 4   ceo_woman                           1000 non-null   int64
 5   Market Cap                          1000 non-null   float64
 6   rank_change                         1000 non-null   float64
 7   state_FL                            1000 non-null   bool
 8   state_GA                            1000 non-null   bool
 9   state_MA                            1000 non-null   bool
 10  state_MI                            1000 non-null   bool
 11  state_MN                            1000 non-null   bool
 12  state_NY                            1000 non-null   bool
 13  state_OH                            1000 non-null   bool
 14  state_PA                            1000 non-null   bool
 15  state_TX                            1000 non-null   bool
 16  state_VA                            1000 non-null   bool
 17  state_state_other                   1000 non-null   bool
 18  city_Atlanta                        1000 non-null   bool
 19  city_Austin                         1000 non-null   bool
...
 65  sector_others                       1000 non-null   bool
 66  profitable                          1000 non-null   int64
```

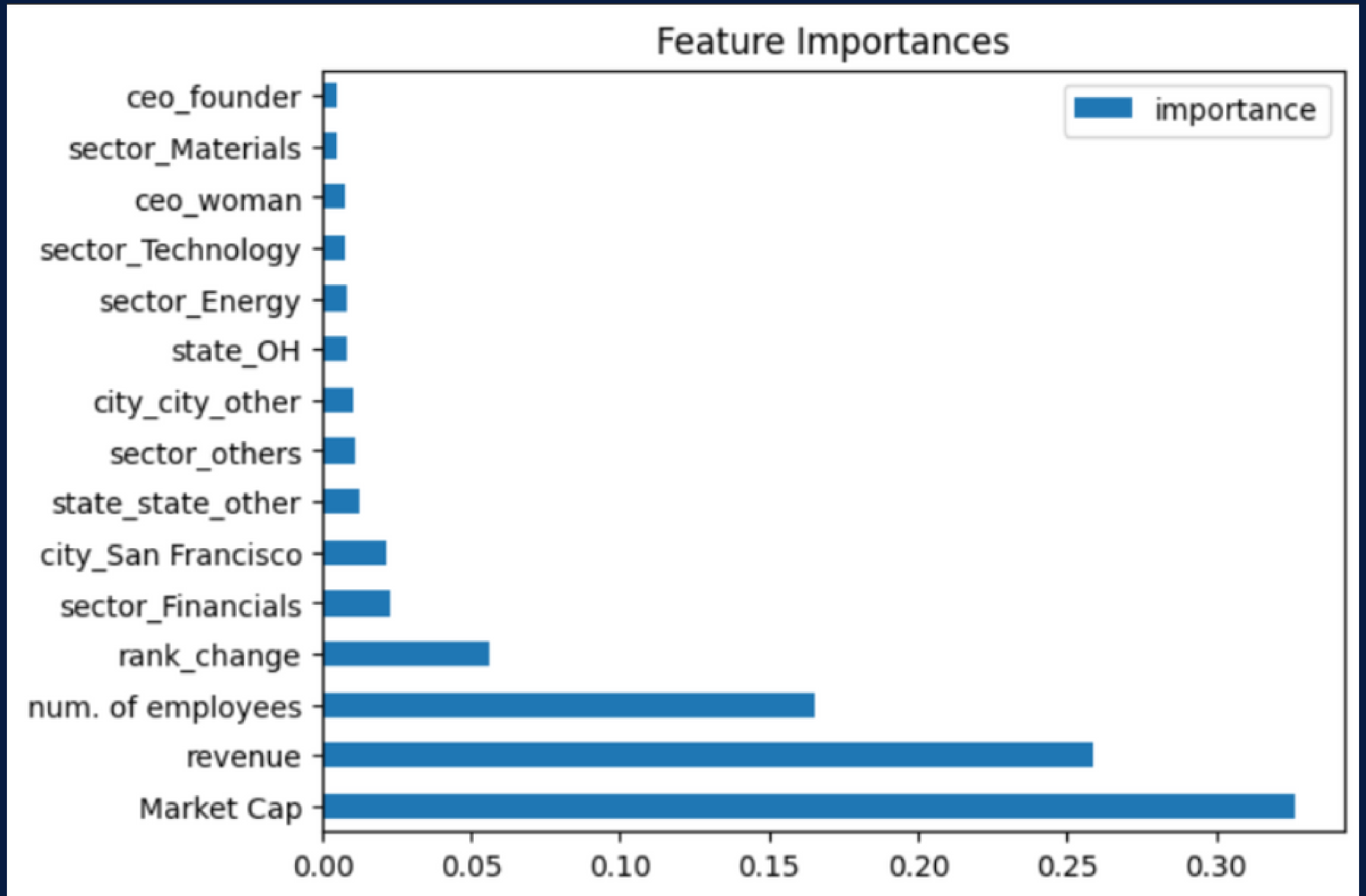
Removing 3% of outliers from train set

```
# Removing outliers with Isolation Forest
outlier_detector = IsolationForest(contamination=0.03)
outlier_detector.fit(X_train)
no_outliers = outlier_detector.predict(X_train)
no_outliers = no_outliers == 1
X_train, y_train = X_train[no_outliers], y_train[no_outliers]
print("Training set - X:", X_train.shape, "y:", y_train.shape)
```

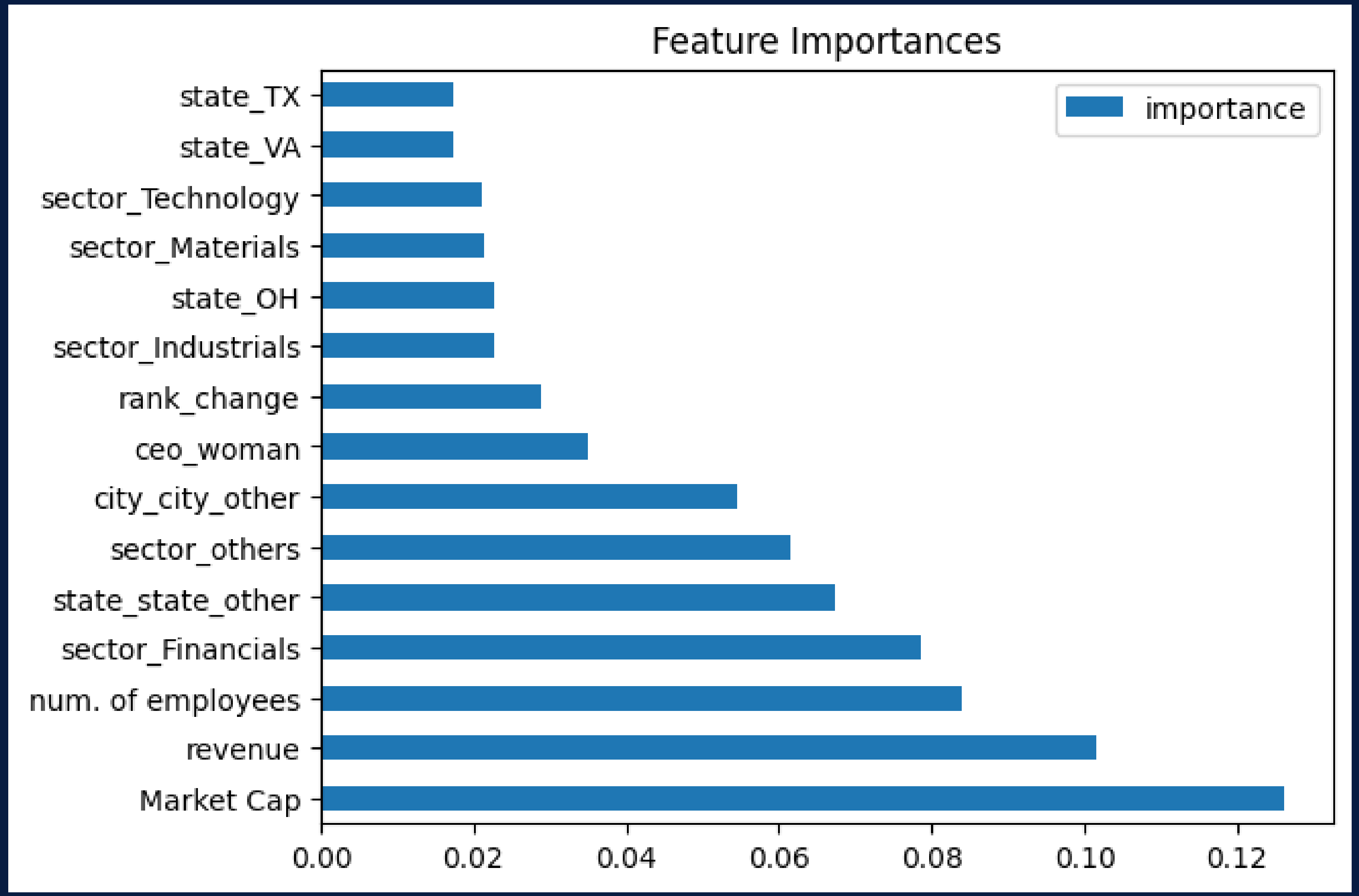
```
Training set - X: (776, 66) y: (776,)
```

SMOTEENN

imblearn.combine



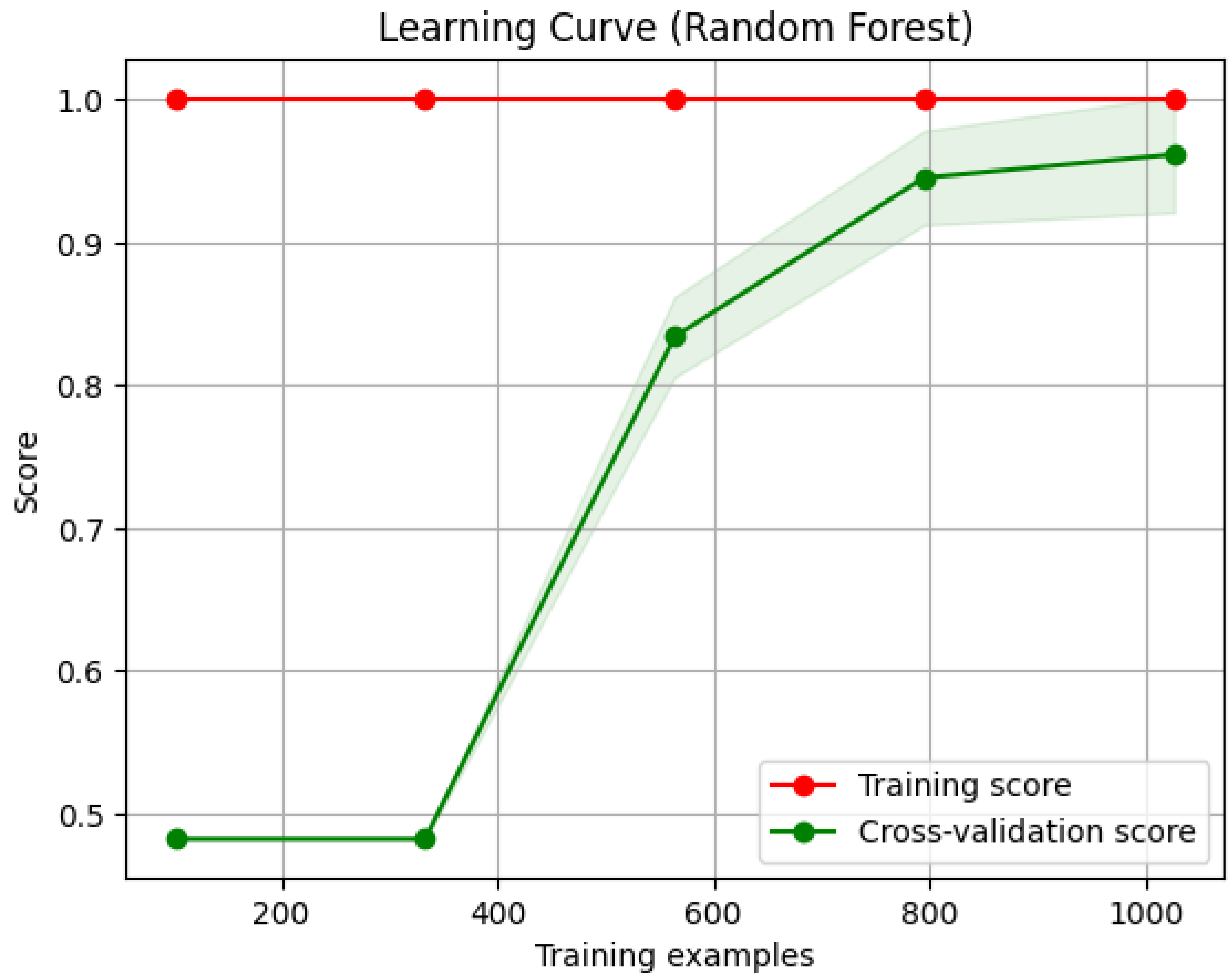
CRUCIO



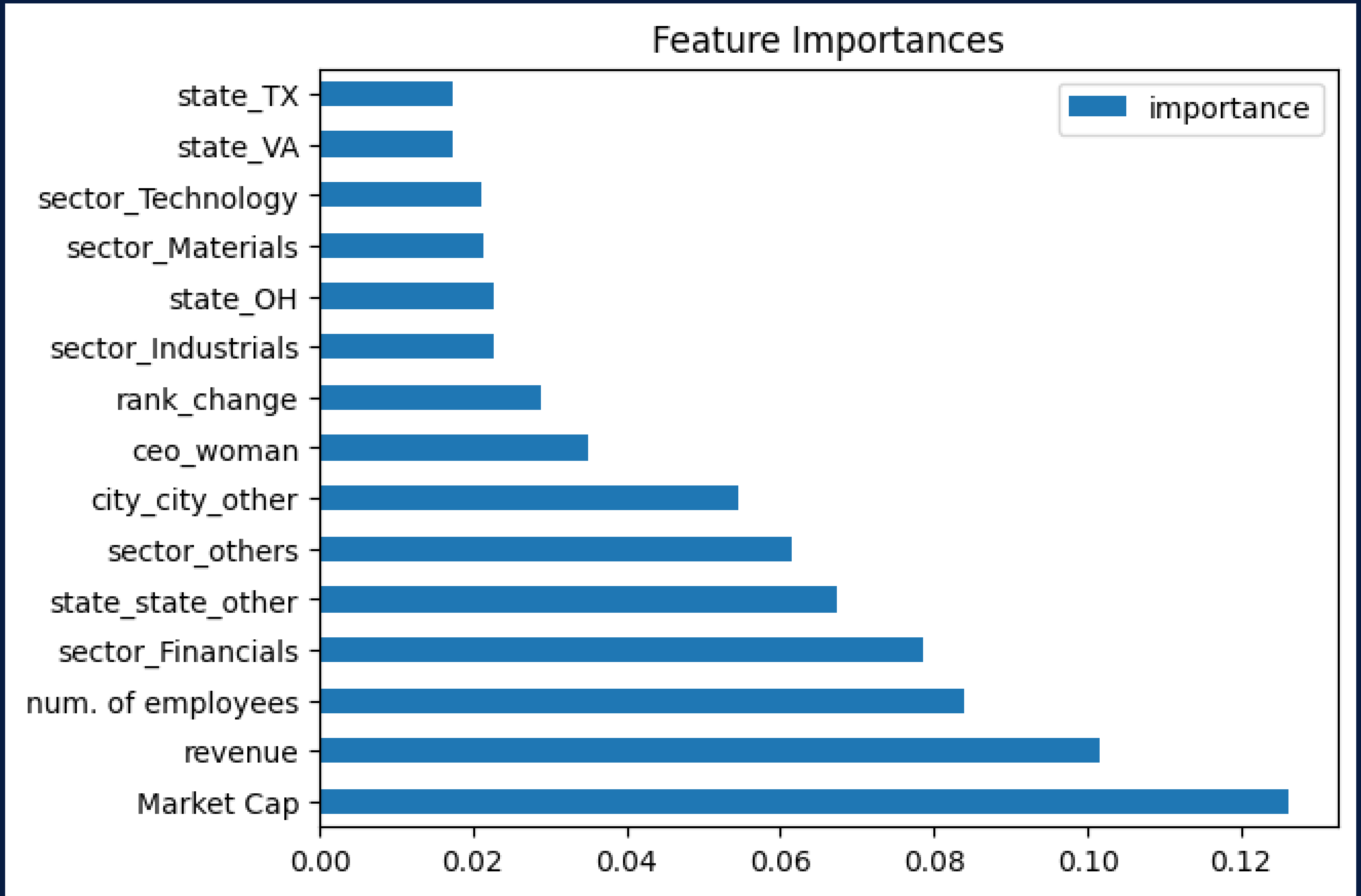
Data Processing

Random Forest

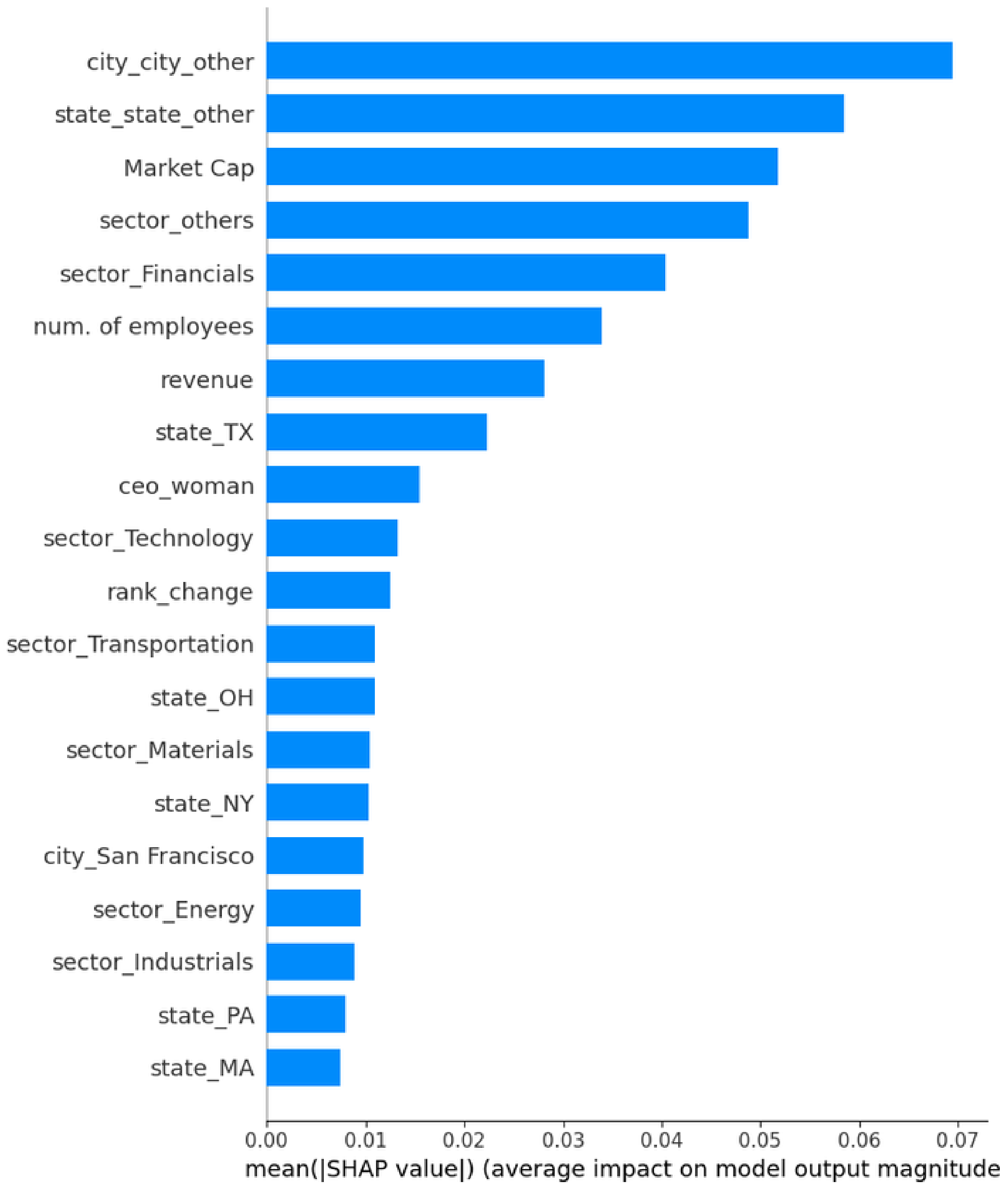
```
Accuracy: 0.88
Precision: 0.8974358974358975
Recall: 0.9776536312849162
F1: 0.9358288770053476
[[  1  20]
 [  4 175]]
```

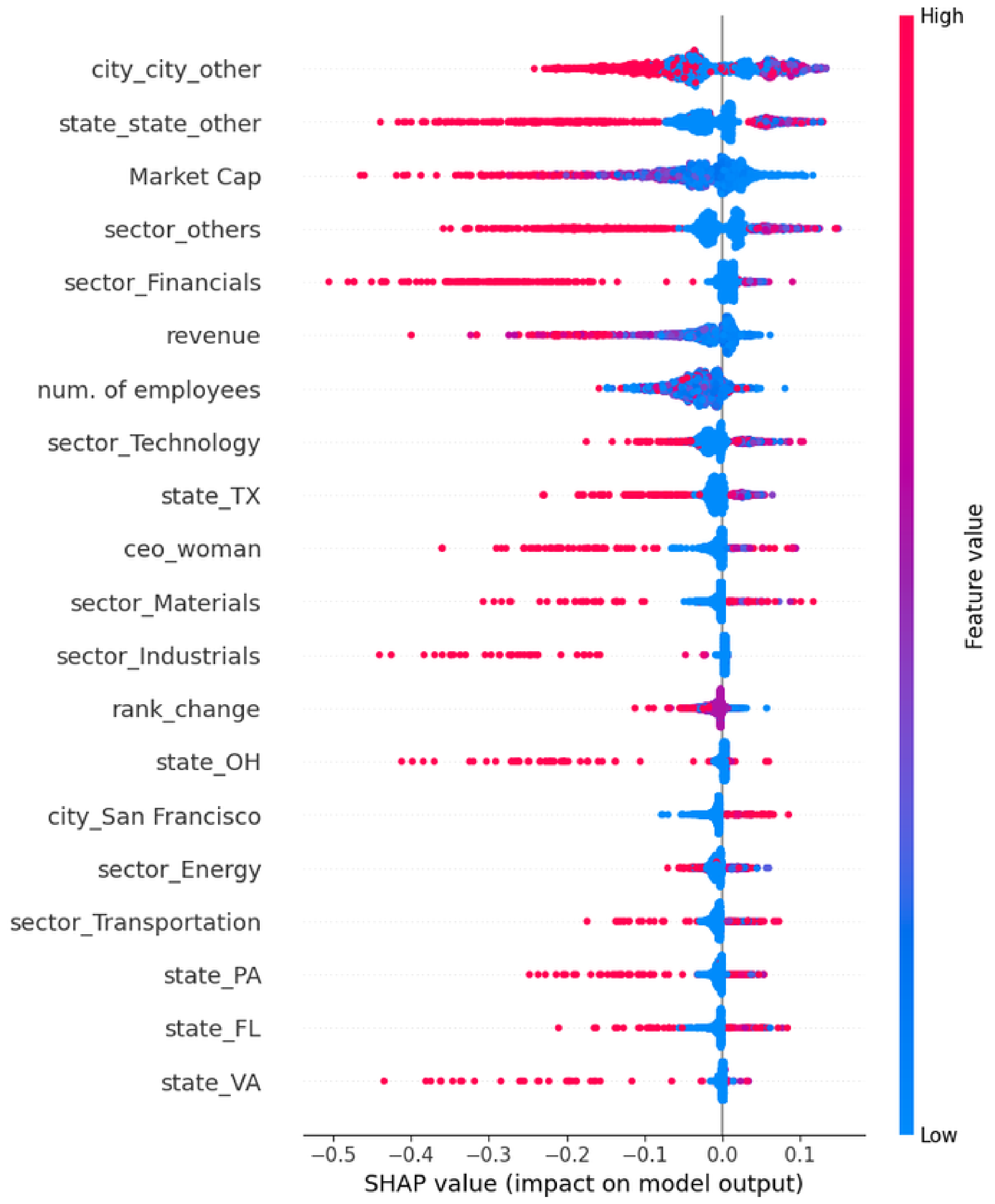


Feature Importances



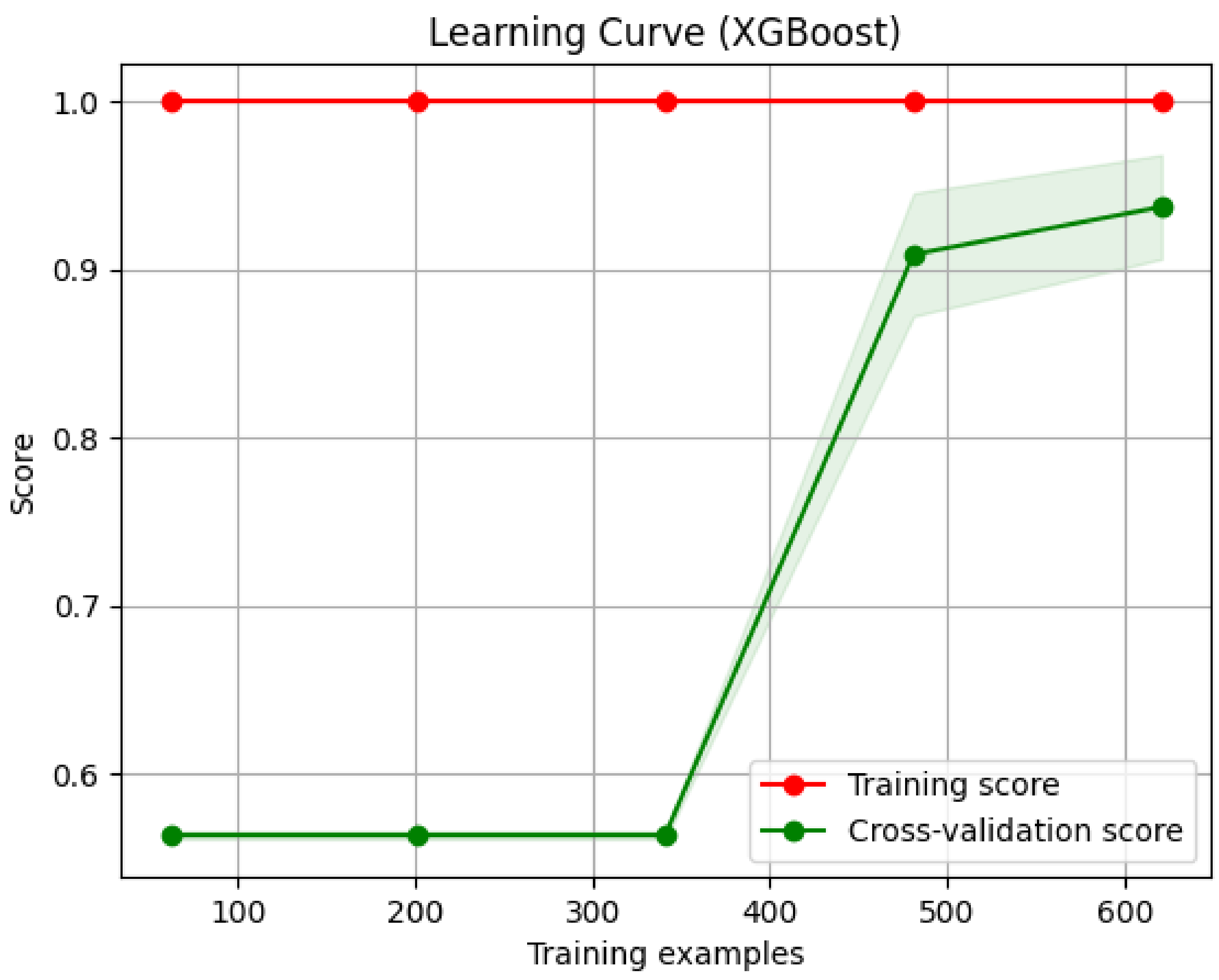
SHAP Plots


city_city_other
state_state_other
Market Cap
sector_others
sector_Financials
num. of employees
revenue
state_TX
ceo_woman
sector_Technology
rank_change
sector_Transportation
state_OH
sector_Materials
state_NY
city_San Francisco
sector_Energy
sector_Industrials
state_PA
state_MA
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
mean(|SHAP value|) (average impact on model output magnitude


city_city_other
state_state_other
Market Cap
sector_others
sector_Financials
revenue
num. of employees
sector_Technology
state_TX
ceo_woman
sector_Materials
sector_Industrials
rank_change
state_OH
city_San Francisco
sector_Energy
sector_Transportation
state_PA
state_FL
state_VA
−0.5
−0.4
−0.3
−0.2
−0.1
0.0
0.1
SHAP value (impact on model output)
High
Low
Feature value

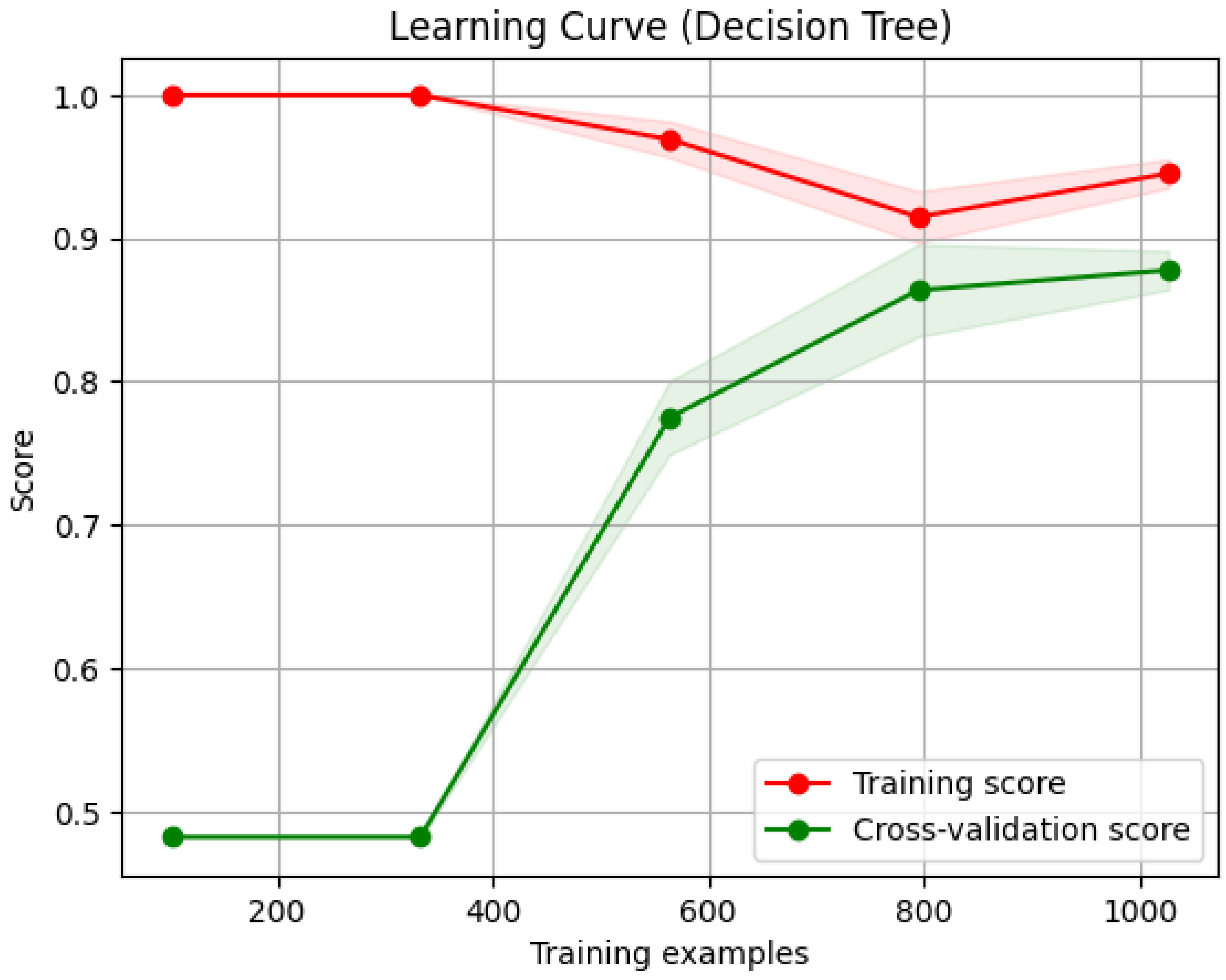
XGBoost

```
Accuracy: 0.77
Precision: 0.9182389937106918
Recall: 0.8156424581005587
F1: 0.8639053254437871
[[  8  13]
 [ 33 146]]
```

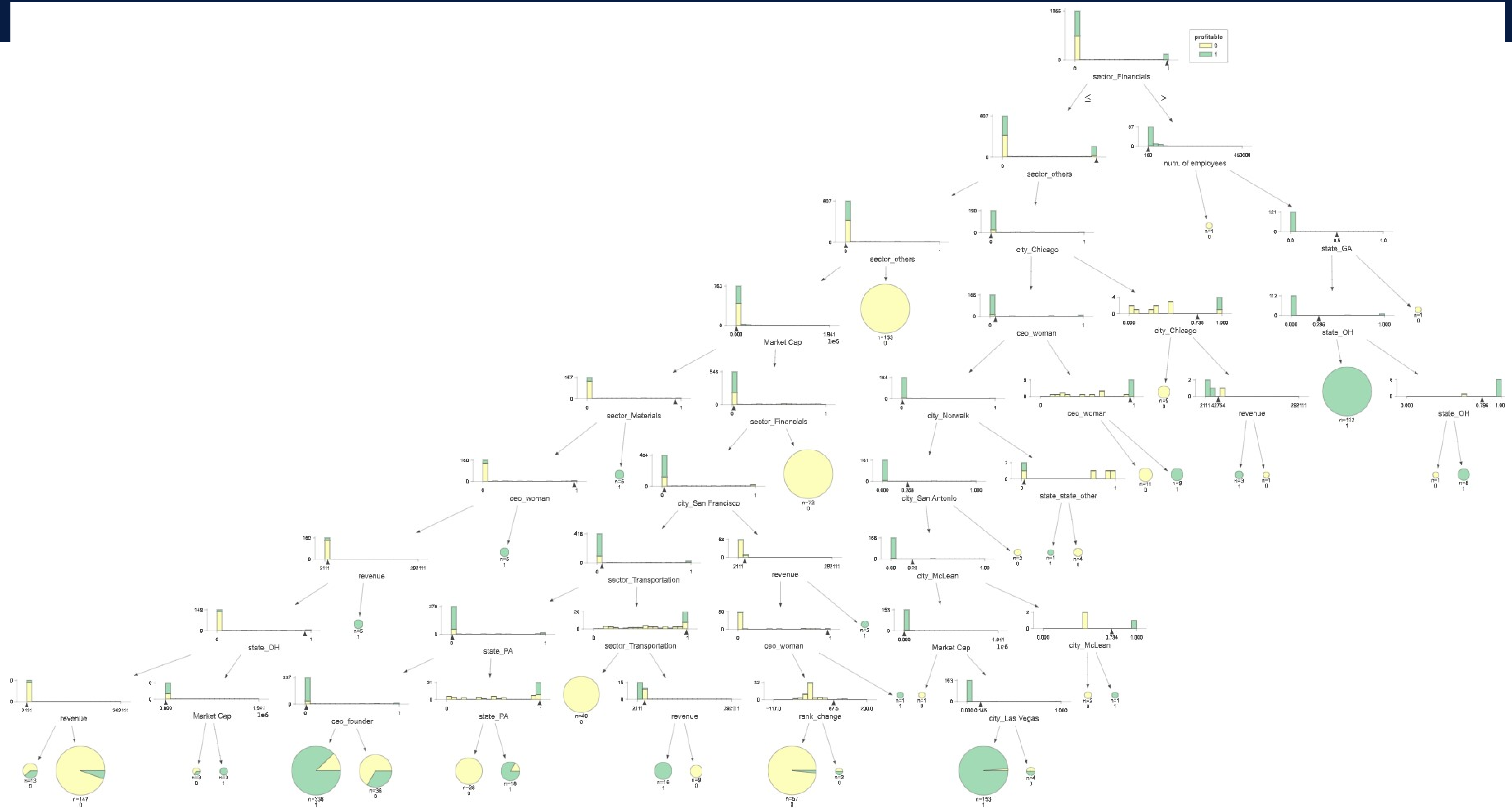


DecisionTreeClassifier

```
Accuracy: 0.84
Precision: 0.9152542372881356
Recall: 0.9050279329608939
F1: 0.9101123595505618
[[  6  15]
 [ 17 162]]
```



Tree visualization



Scaling data

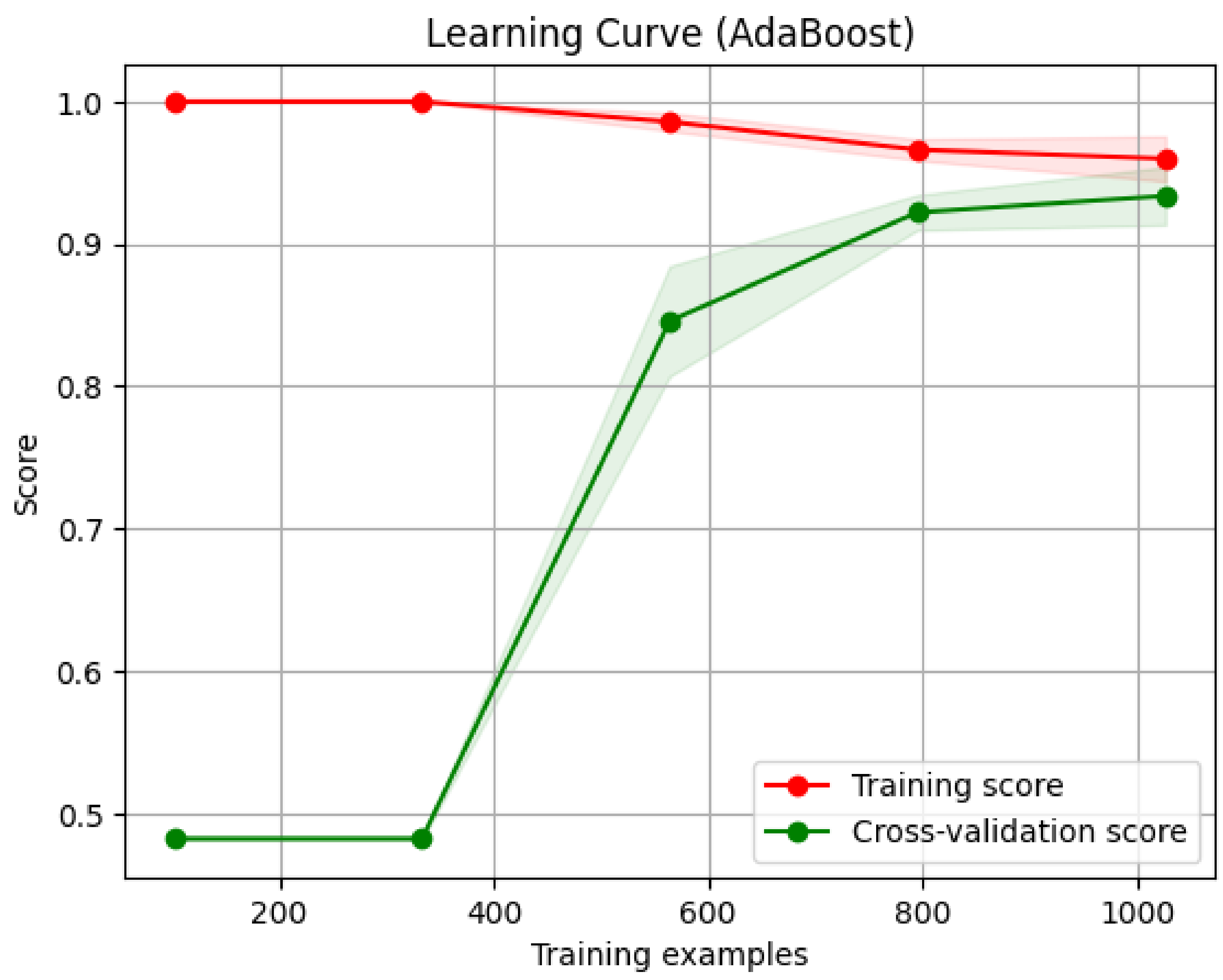
```
#Normalize the data
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Logistic Regression

```
Accuracy: 0.82
Precision: 0.9181286549707602
Recall: 0.8770949720670391
F1: 0.8971428571428571
[[  7  14]
 [ 22 157]]
```

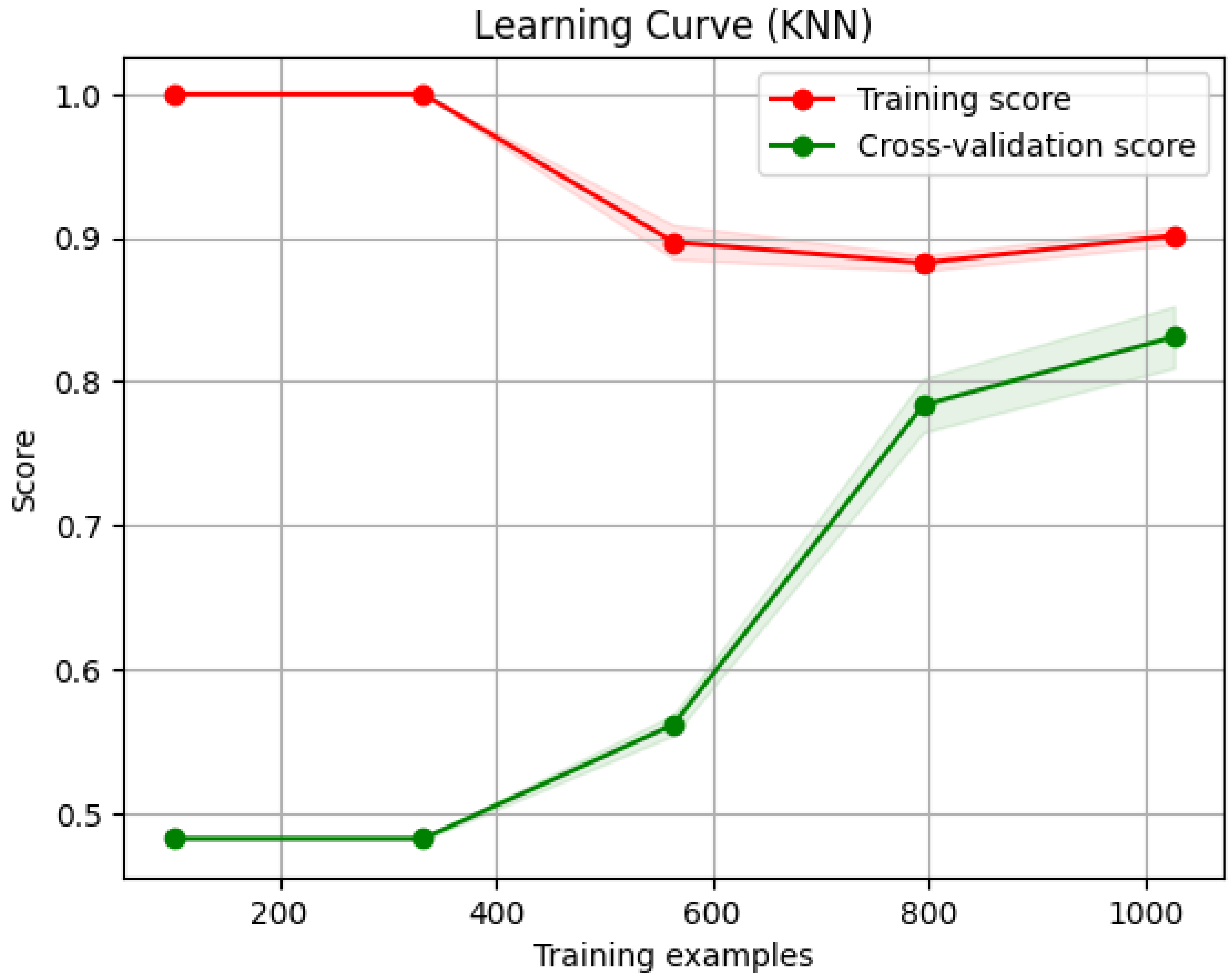
AdaBoost

```
Accuracy: 0.855
Precision: 0.9166666666666666
Recall: 0.9217877094972067
F1: 0.9192200557103063
[[  6  15]
 [ 14 165]]
```

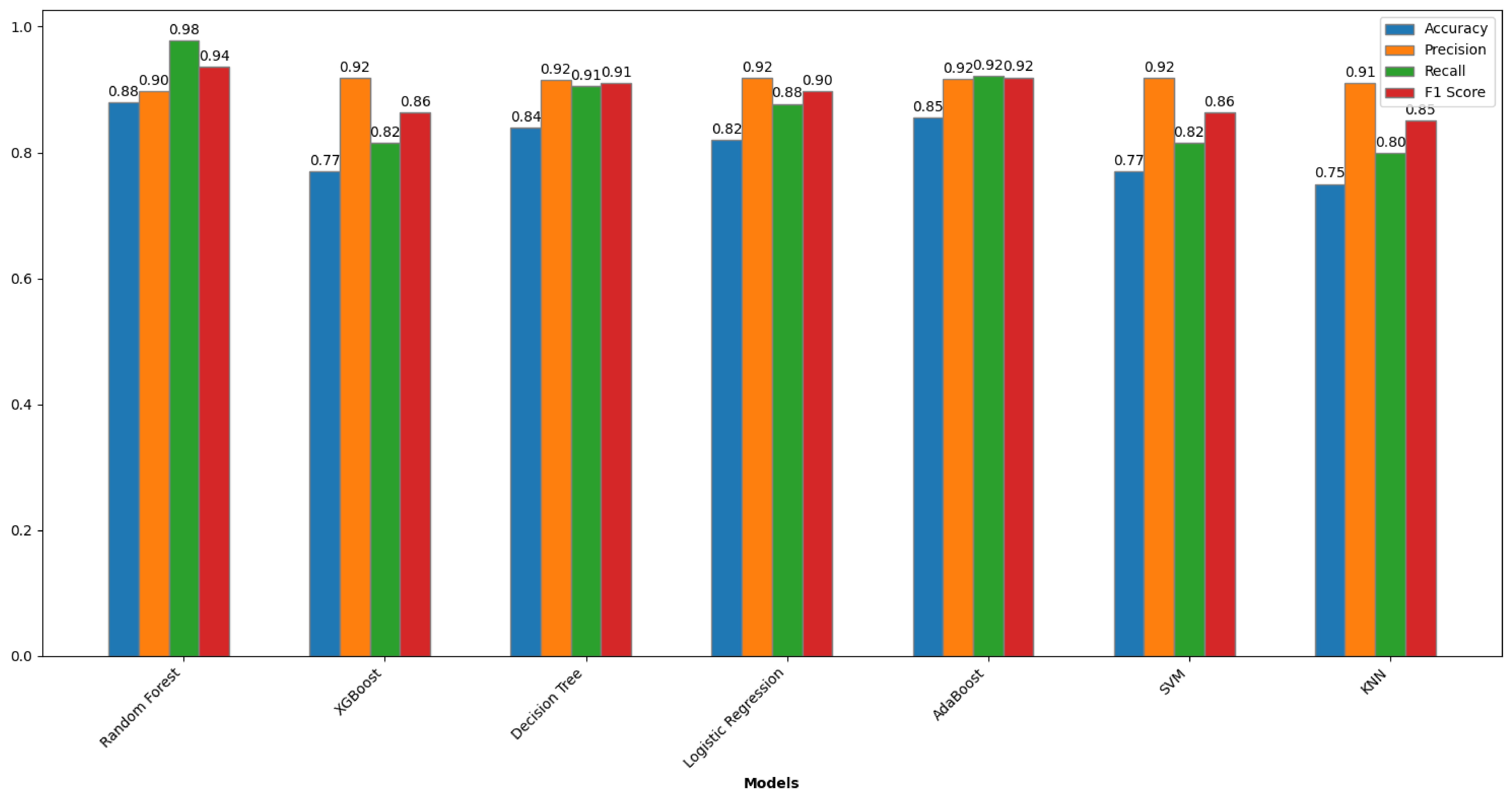


KNN

```
Accuracy: 0.75
Precision: 0.910828025477707
Recall: 0.7988826815642458
F1: 0.8511904761904762
[[  7  14]
 [ 36 143]]
```



Models performance




THANK YOU
FOR YOUR
ATTENTION