Mercedes-Benz Greener Manufacturing

Importing Library

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import preprocessing
import xgboost as xgb
color = sns.color_palette()

%matplotlib inline

pd.options.mode.chained_assignment = None  # default='warn'
pd.options.display.max_columns = 999

from subprocess import check_output
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning:
  import pandas.util.testing as tm
```

```
print(check_output(["ls"]).decode("utf8"))
```

```
sample_data
test.csv
train.csv
```

Importing Dataset

```
train_df = pd.read_csv("train.csv")
test_df = pd.read_csv("test.csv")
print("Train shape : ", train_df.shape)
print("Test shape : ", test_df.shape)
```

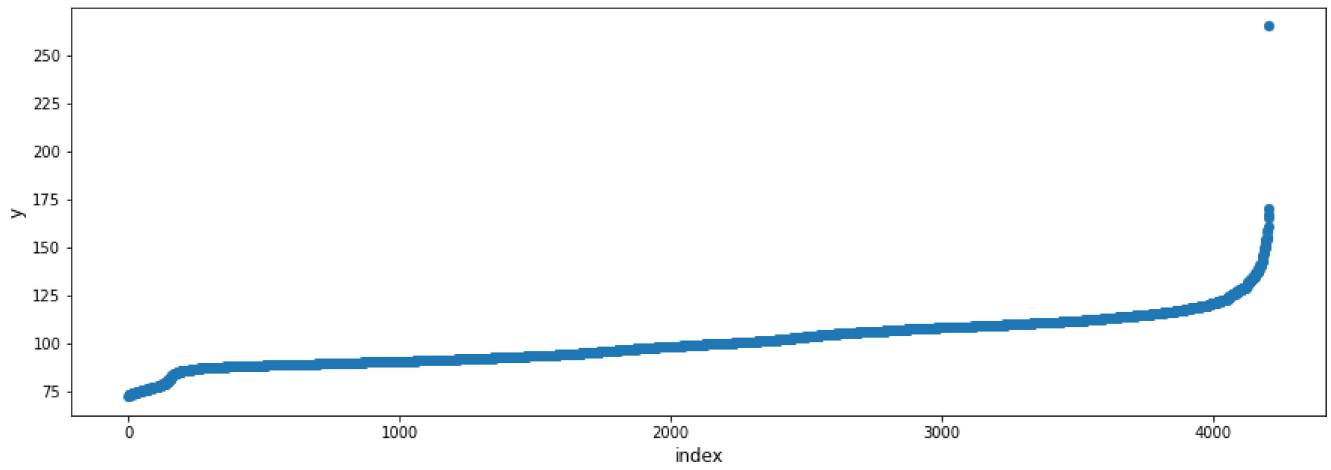
```
Train shape :  (4209, 378)
Test shape :  (4209, 377)
```

```
train_df.head()
```

```
train_df.groupby("X0").aggregate('count').reset_index()
```

	ID	y	X0	X1	X2	X3	X4	X5	X6	X8	X10	X11	X12	X13	X14	X15	X16	X17	X
0	0	130.81	k	v	at	a	d	u	j	o	0	0	0	1	0	0	0	0	
1	6	88.53	k	t	av	e	d	y	l	o	0	0	0	0	0	0	0	0	
2	7	76.26	az	w	n	c	d	x	j	x	0	0	0	0	0	0	0	1	
3	9	80.62	az	t	n	f	d	x	l	e	0	0	0	0	0	0	0	0	
4	13	78.02	az	v	n	f	d	h	d	n	0	0	0	0	0	0	0	0	

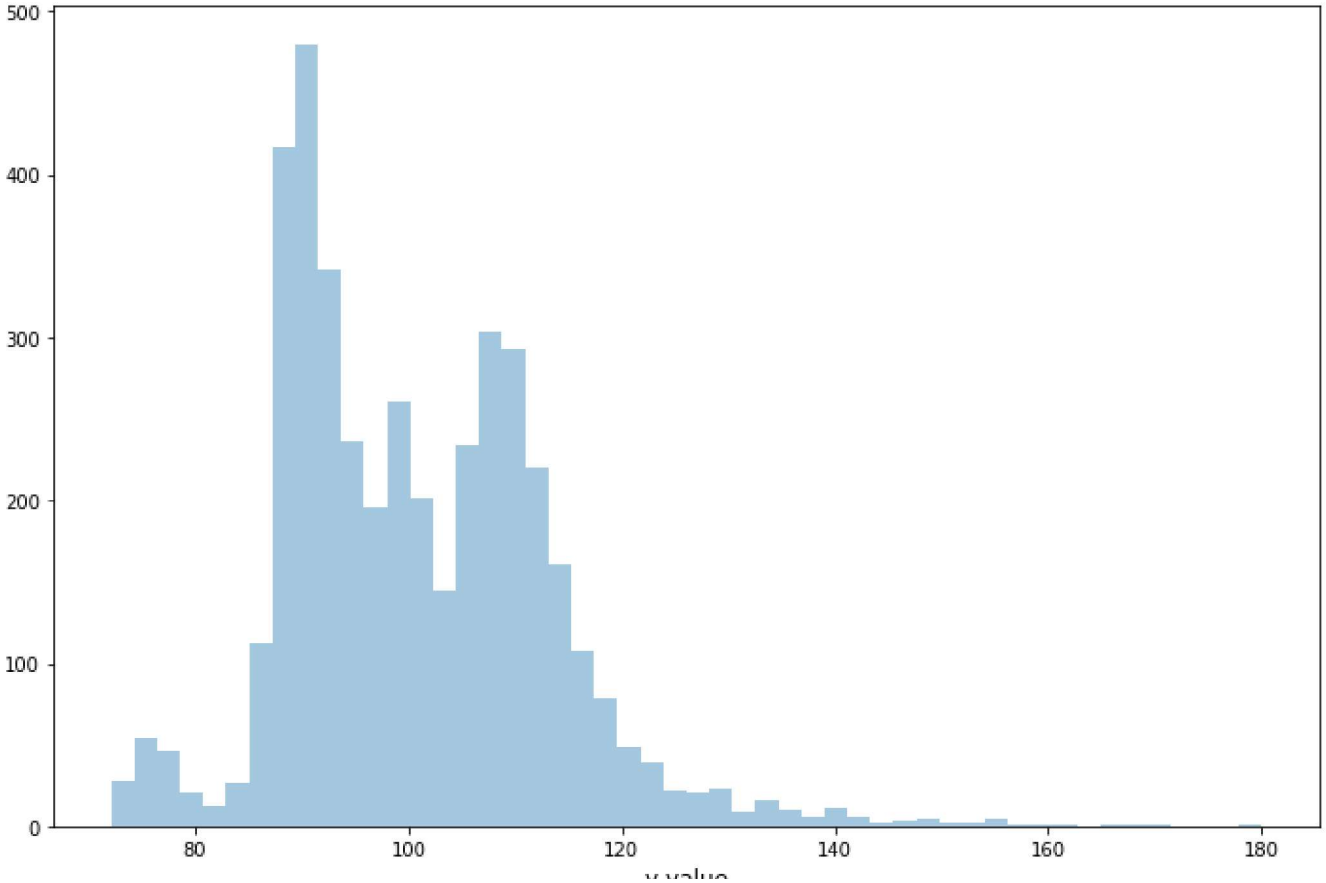
```
plt.figure(figsize=(15,5))
plt.scatter(range(train_df.shape[0]), np.sort(train_df.y.values))
plt.xlabel('index', fontsize=12)
plt.ylabel('y', fontsize=12)
plt.show()
```




```
train_df['y'].max()
```

```
265.32
```

```
ulimit = 180
train_df['y'].loc[train_df['y']>ulimit] = ulimit

plt.figure(figsize=(12,8))
sns.distplot(a=train_df.y.values, bins=50, kde=False)
plt.xlabel('y value', fontsize=12)
plt.show()
```



Data_Type For each Column

```
dtype_df = train_df.dtypes.reset_index()
dtype_df.columns = ["Count", "Column Type"]
dtype_df.groupby("Column Type").aggregate('count').reset_index()
```

	Column Type	Count
0	int64	369
1	float64	1
2	object	8

```
dtype_df.loc[:10,:]
```

	Count	Column Type
0	ID	int64
1	y	float64
2	X0	object
3	X1	object
4	X2	object
5	X3	object

Missing values

```
missing_df = train_df.isnull().sum(axis=0).reset_index()
missing_df.columns = ['column_name', 'missing_count']
missing_df = missing_df.loc[missing_df['missing_count']>0]
missing_df = missing_df.sort_values(by='missing_count')
missing_df
```

	column_name	missing_count

Integer Columns

```
unique_values_dict = {}
for col in train_df.columns:
    if col not in ["ID", "y", "X0", "X1", "X2", "X3", "X4", "X5", "X6", "X8"]:
        unique_value = str(np.sort(train_df[col].unique()).tolist())
        tlist = unique_values_dict.get(unique_value,[])
        tlist.append(col)
        unique_values_dict[unique_value] = tlist[:]
for unique_val, columns in unique_values_dict.items():
    print("Columns containing the unique values : ",unique_val)
    print(columns)
    print("--------------------------------------------------")
```

```
Columns containing the unique values :  [0, 1]
['X10', 'X12', 'X13', 'X14', 'X15', 'X16', 'X17', 'X18', 'X19', 'X20', 'X21', 'X22', 'X
--------------------------------------------------
Columns containing the unique values :  [0]
['X11', 'X93', 'X107', 'X233', 'X235', 'X268', 'X289', 'X290', 'X293', 'X297', 'X330',
--------------------------------------------------
```

Categorical Columns

The number of 0's and 1's in each of these variables.

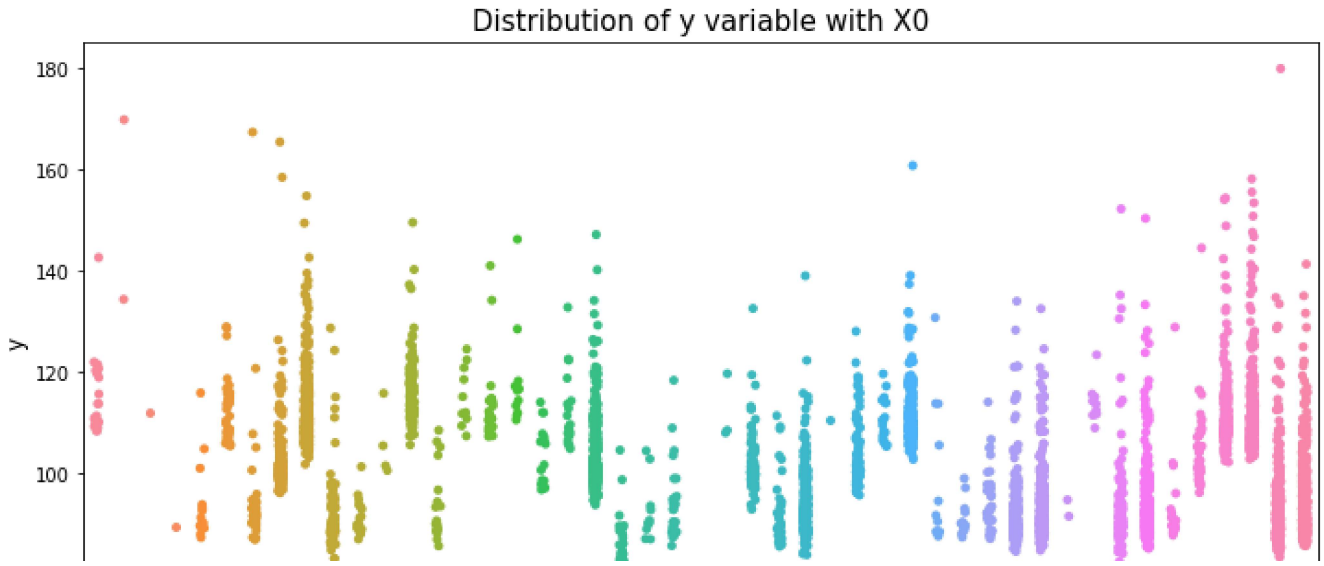
```
zero_count_list = []
one_count_list = []
cols_list = unique_values_dict['[0, 1]']
for col in cols_list:
    zero_count_list.append((train_df[col]==0).sum())
    one_count_list.append((train_df[col]==1).sum())
N = len(cols_list)
ind = np.arange(N)


width = 0.35
plt.figure(figsize=(6,100))
p1 = plt.barh(ind, zero_count_list, width, color='red')
p2 = plt.barh(ind, one_count_list, width, left=zero_count_list, color="blue")
plt.yticks(ticks=ind,labels=cols_list)
plt.legend((p1[0], p2[0]), ('Zero count', 'One Count'))
plt.show()
```

```
            lbl.fit(list(train_df[f].values))
            train_df[f] = lbl.transform(list(train_df[f].values))

    train_y = train_df['y'].values
    train_X = train_df.drop(["ID", "y", "eval_set"], axis=1)

    # Thanks to anokas for this #
    def xgb_r2_score(preds, dtrain):
        labels = dtrain.get_label()
        return 'r2', r2_score(labels, preds)

    xgb_params = {
        'eta': 0.05,
        'max_depth': 6,
        'subsample': 0.7,
        'colsample_bytree': 0.7,
        'objective': 'reg:linear',
        'silent': 1
    }
    dtrain = xgb.DMatrix(train_X, train_y, feature_names=train_X.columns.values)
    model = xgb.train(dict(xgb_params, silent=0), dtrain, num_boost_round=100, feval=xgb_r2_score

    # plot the important features #
    fig, ax = plt.subplots(figsize=(12,18))
    xgb.plot_importance(model, max_num_features=50, height=0.8,ax=ax)
    plt.show()
```

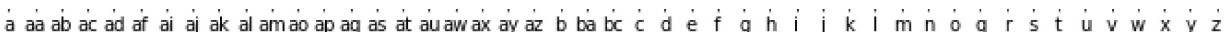
	X0	ID	y	X1	X2	X3	X4	X5	X6	X8	X10	X11	X12	X13	X14	X15	X16
0	a	21	21	21	21	21	21	21	21	21	21	21	21	21	21	21	21
1	aa	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
2	ab	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
3	ac	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
4	ad	14	14	14	14	14	14	14	14	14	14	14	14	14	14	14	14
5	af	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35
6	ai	34	34	34	34	34	34	34	34	34	34	34	34	34	34	34	34
7	aj	151	151	151	151	151	151	151	151	151	151	151	151	151	151	151	151
8	ak	349	349	349	349	349	349	349	349	349	349	349	349	349	349	349	349
9	al	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67
10	am	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18
11	ao	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
12	ap	103	103	103	103	103	103	103	103	103	103	103	103	103	103	103	103
13	aq	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18
14	as	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
15	at	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25
16	au	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11
17	aw	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16
18	ax	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19
20	az	175	175	175	175	175	175	175	175	175	175	175	175	175	175	175	175

column -> X0

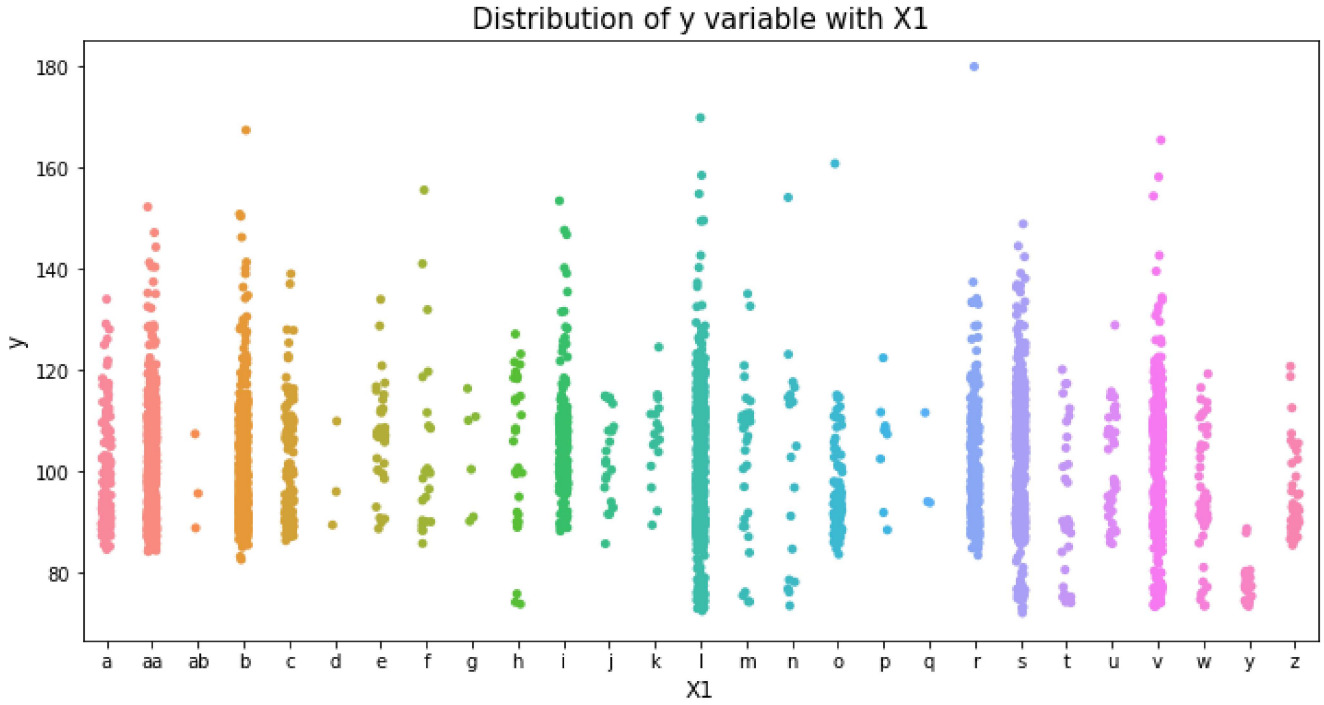
```
var_name = "X0"
col_order = np.sort(train_df[var_name].unique()).tolist()
plt.figure(figsize=(12,6))
sns.stripplot(x=var_name, y='y', data=train_df, order=col_order)
plt.xlabel(var_name, fontsize=12)
plt.ylabel('y', fontsize=12)
plt.title("Distribution of y variable with "+var_name, fontsize=15)
plt.show()
```



column -> X1

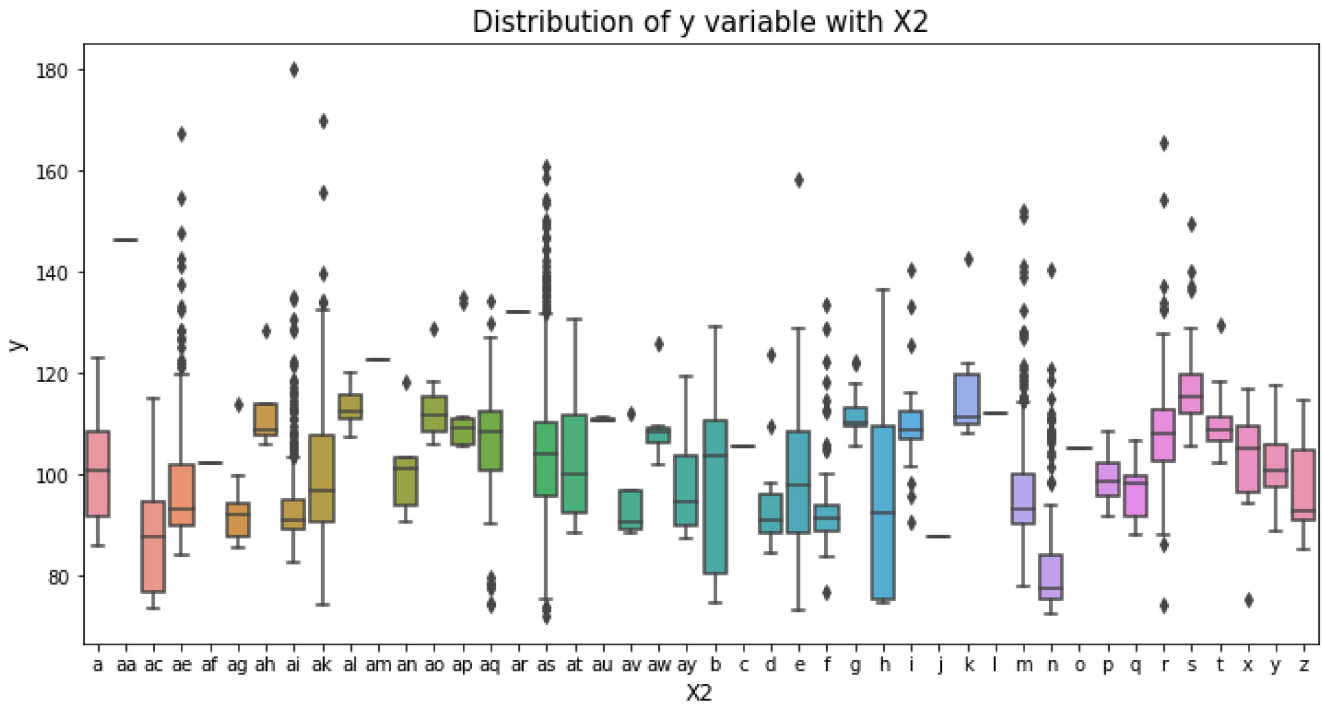



```
var_name = "X1"
col_order = np.sort(train_df[var_name].unique()).tolist()
plt.figure(figsize=(12,6))
sns.stripplot(x=var_name, y='y', data=train_df, order=col_order)
plt.xlabel(var_name, fontsize=12)
plt.ylabel('y', fontsize=12)
plt.title("Distribution of y variable with "+var_name, fontsize=15)
plt.show()
```



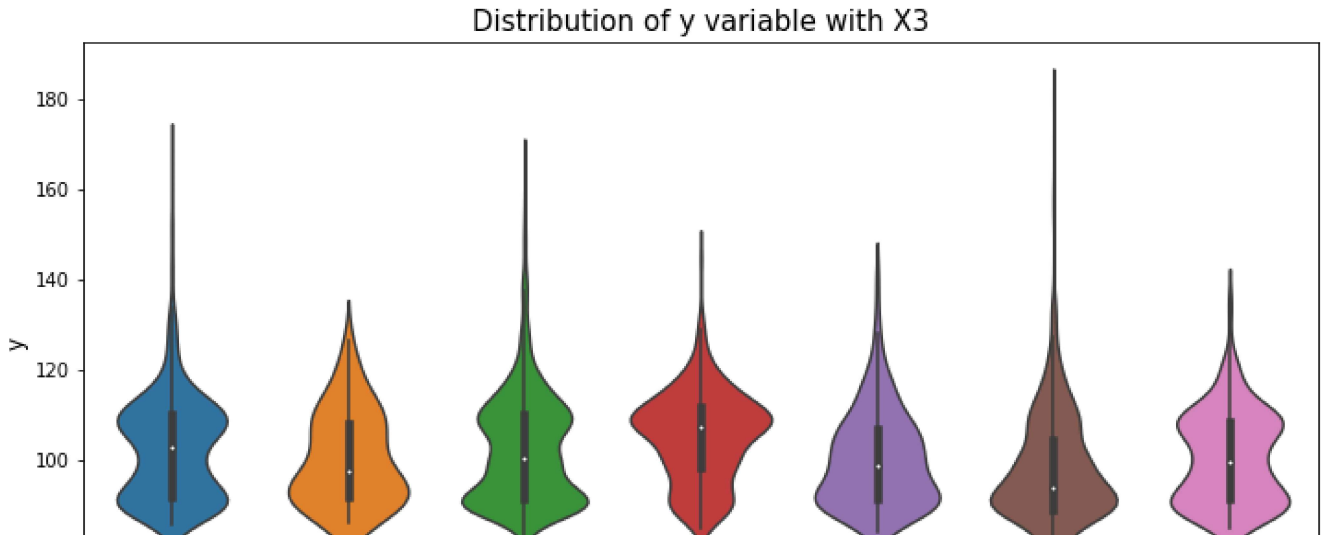
column -> X2

```
var_name = "X2"
col_order = np.sort(train_df[var_name].unique()).tolist()
plt.figure(figsize=(12,6))
sns.boxplot(x=var_name, y='y', data=train_df, order=col_order)
plt.xlabel(var_name, fontsize=12)
plt.ylabel('y', fontsize=12)
plt.title("Distribution of y variable with "+var_name, fontsize=15)
plt.show()
```



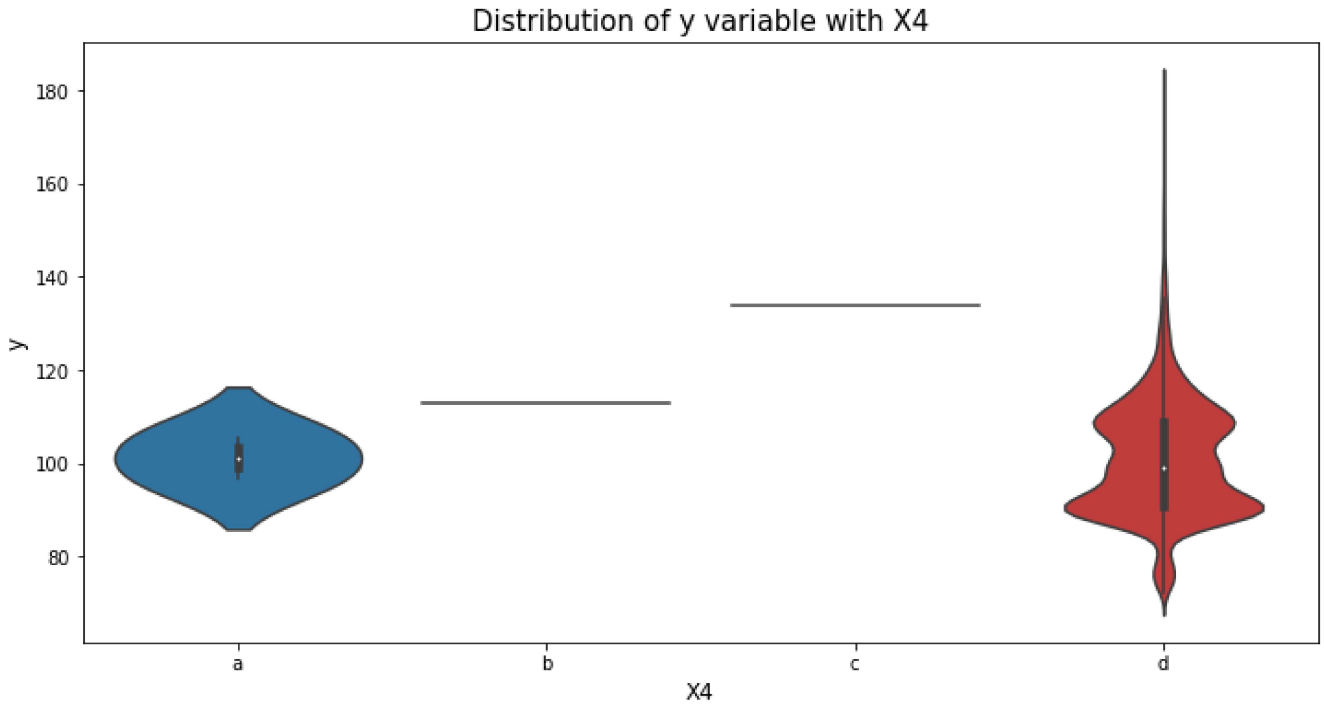
column -> X3

```
var_name = "X3"
col_order = np.sort(train_df[var_name].unique()).tolist()
plt.figure(figsize=(12,6))
sns.violinplot(x=var_name, y='y', data=train_df, order=col_order)
plt.xlabel(var_name, fontsize=12)
plt.ylabel('y', fontsize=12)
plt.title("Distribution of y variable with "+var_name, fontsize=15)
plt.show()
```



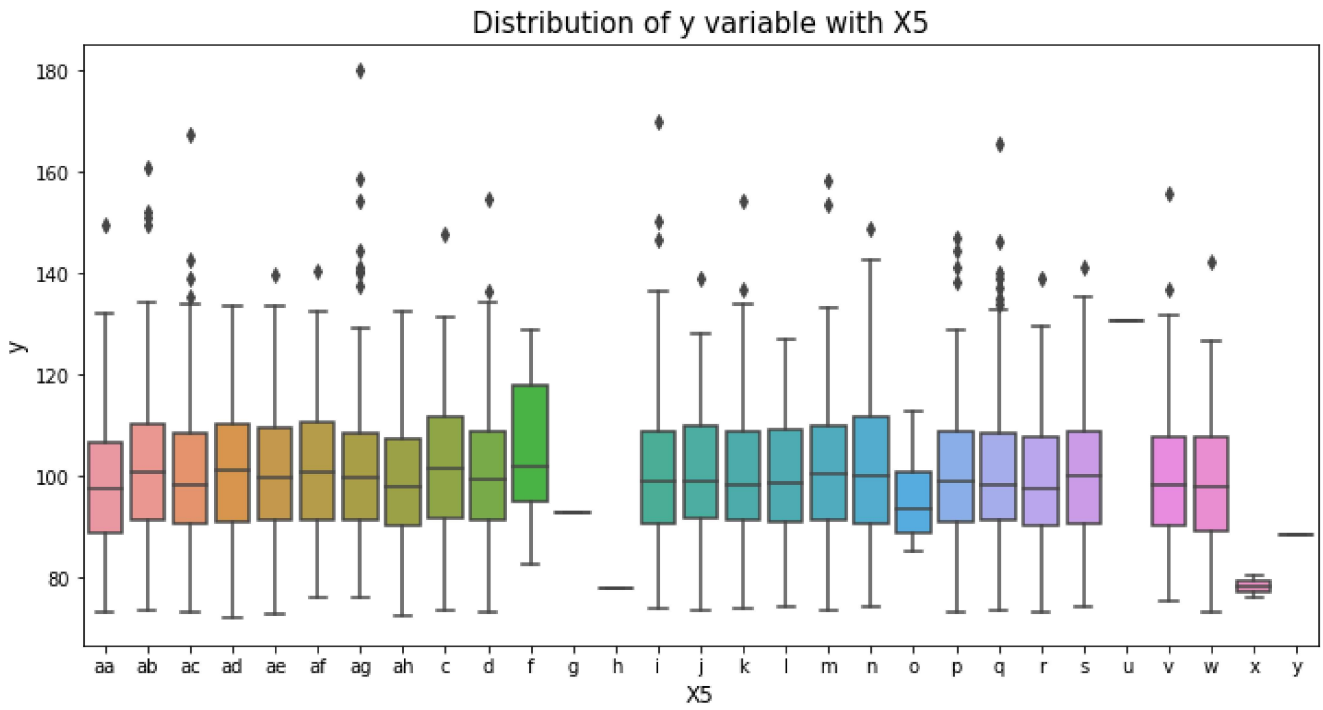
column -> X4


```
var_name = "X4"
col_order = np.sort(train_df[var_name].unique()).tolist()
plt.figure(figsize=(12,6))
sns.violinplot(x=var_name, y='y', data=train_df, order=col_order)
plt.xlabel(var_name, fontsize=12)
plt.ylabel('y', fontsize=12)
plt.title("Distribution of y variable with "+var_name, fontsize=15)
plt.show()
```



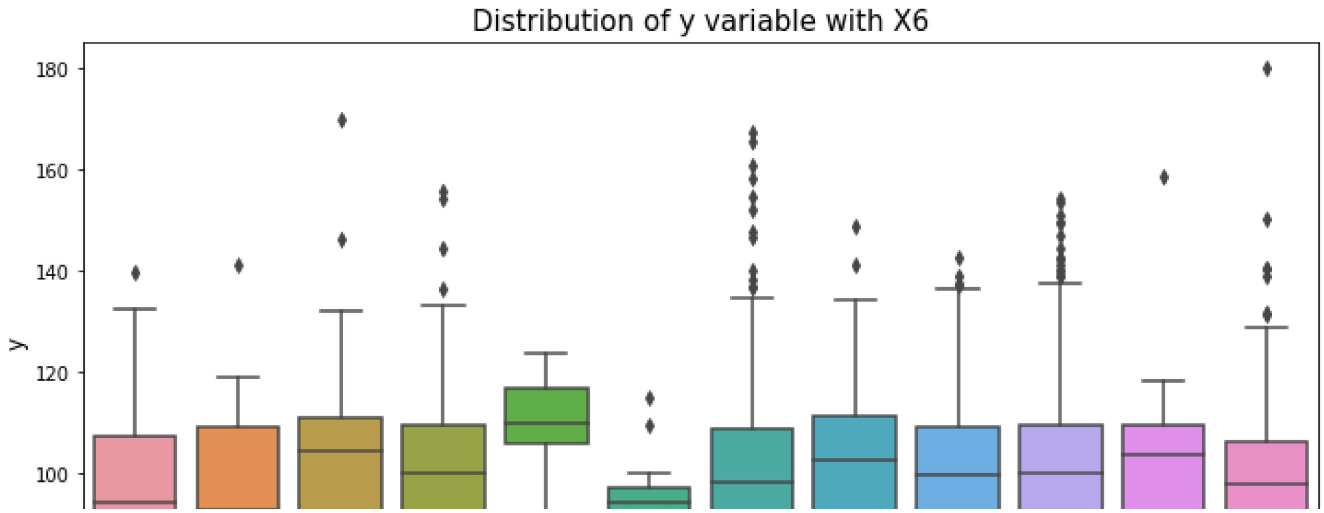
column -> X5

```
var_name = "X5"
col_order = np.sort(train_df[var_name].unique()).tolist()
plt.figure(figsize=(12,6))
sns.boxplot(x=var_name, y='y', data=train_df, order=col_order)
plt.xlabel(var_name, fontsize=12)
plt.ylabel('y', fontsize=12)
plt.title("Distribution of y variable with "+var_name, fontsize=15)
plt.show()
```



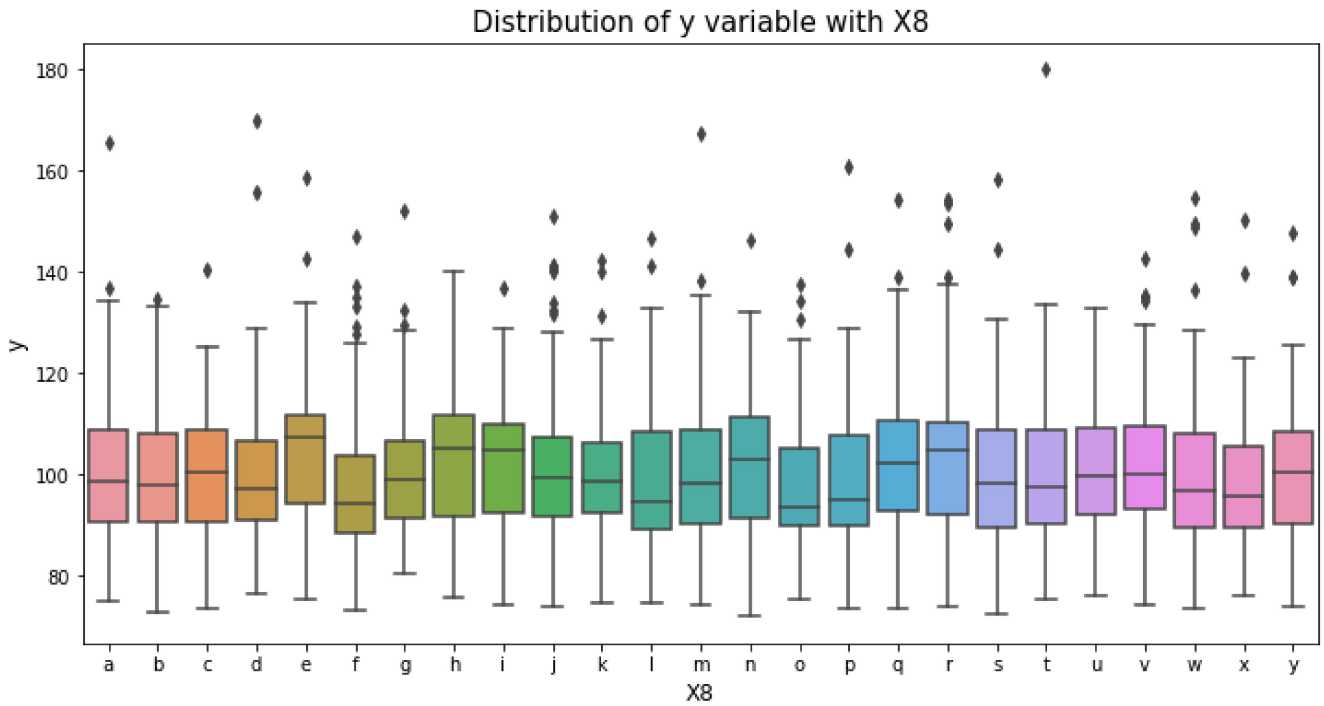
column -> X6

```
var_name = "X6"
col_order = np.sort(train_df[var_name].unique()).tolist()
plt.figure(figsize=(12,6))
sns.boxplot(x=var_name, y='y', data=train_df, order=col_order)
plt.xlabel(var_name, fontsize=12)
plt.ylabel('y', fontsize=12)
plt.title("Distribution of y variable with "+var_name, fontsize=15)
plt.show()
```

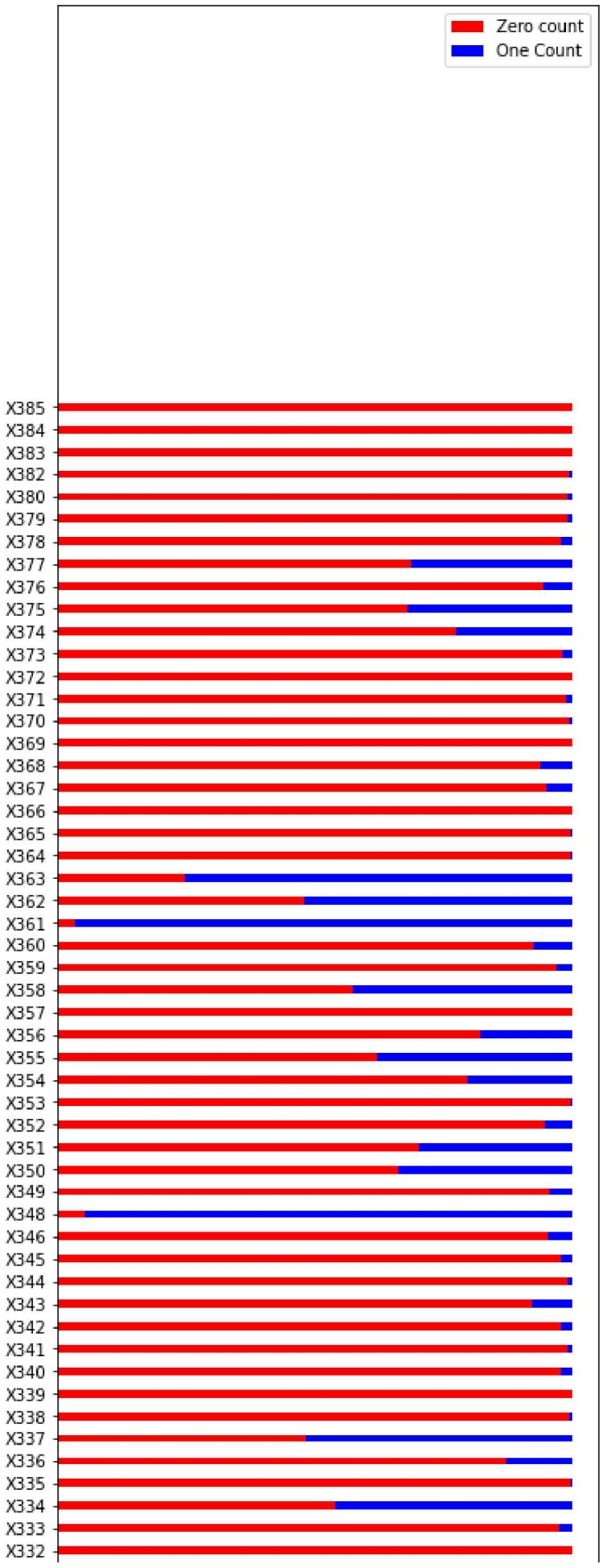


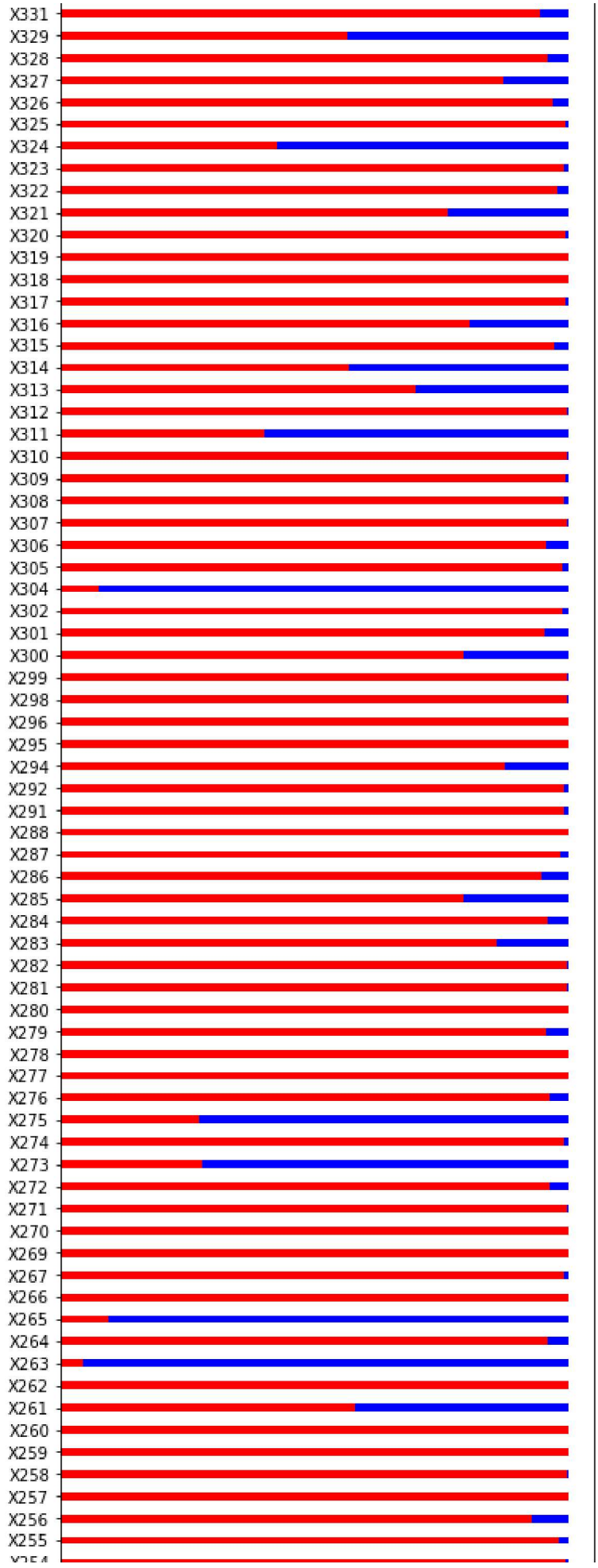
column -> X8

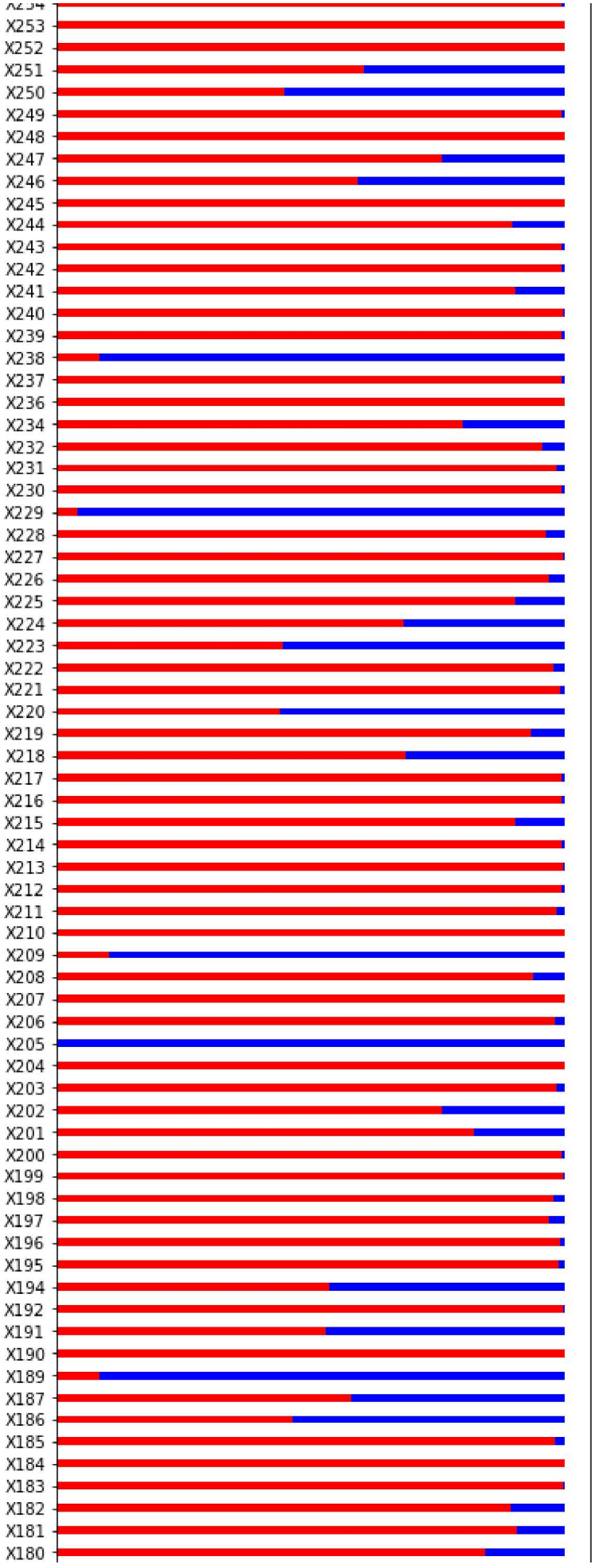
```
var_name = "X8"
col_order = np.sort(train_df[var_name].unique()).tolist()
plt.figure(figsize=(12,6))
sns.boxplot(x=var_name, y='y', data=train_df, order=col_order)
plt.xlabel(var_name, fontsize=12)
plt.ylabel('y', fontsize=12)
plt.title("Distribution of y variable with "+var_name, fontsize=15)
plt.show()
```

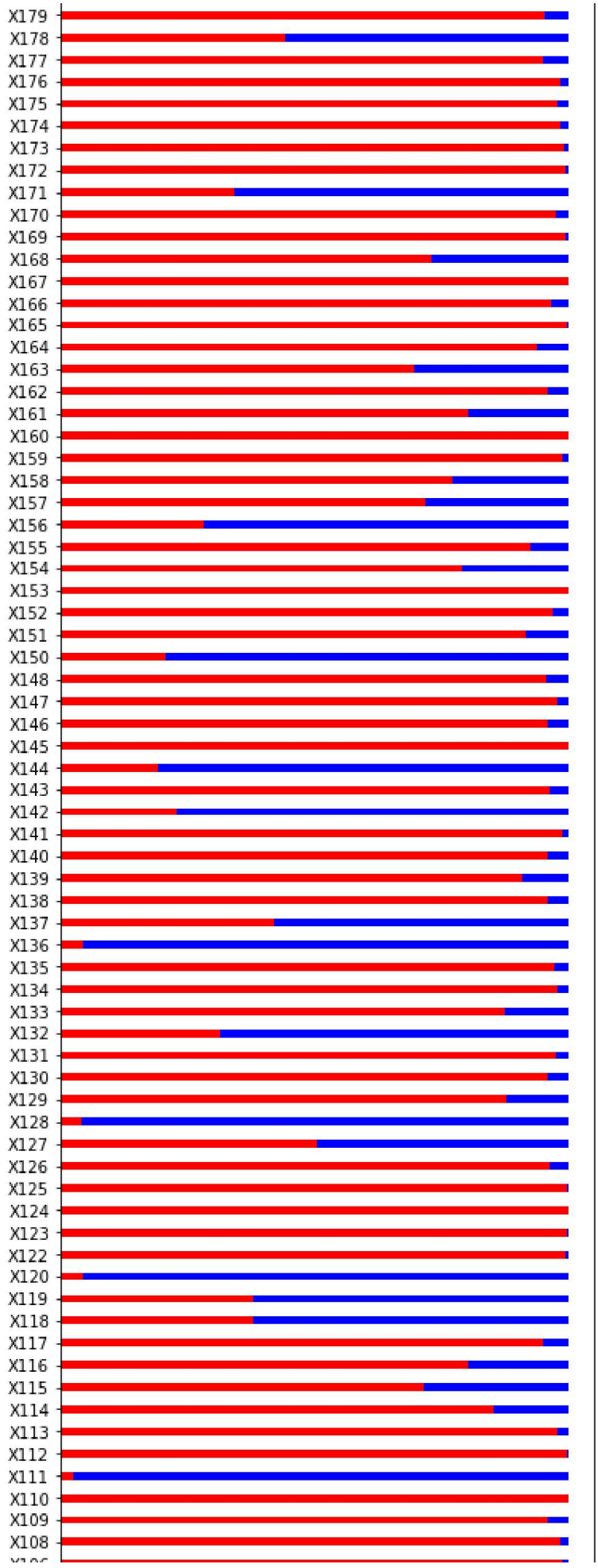


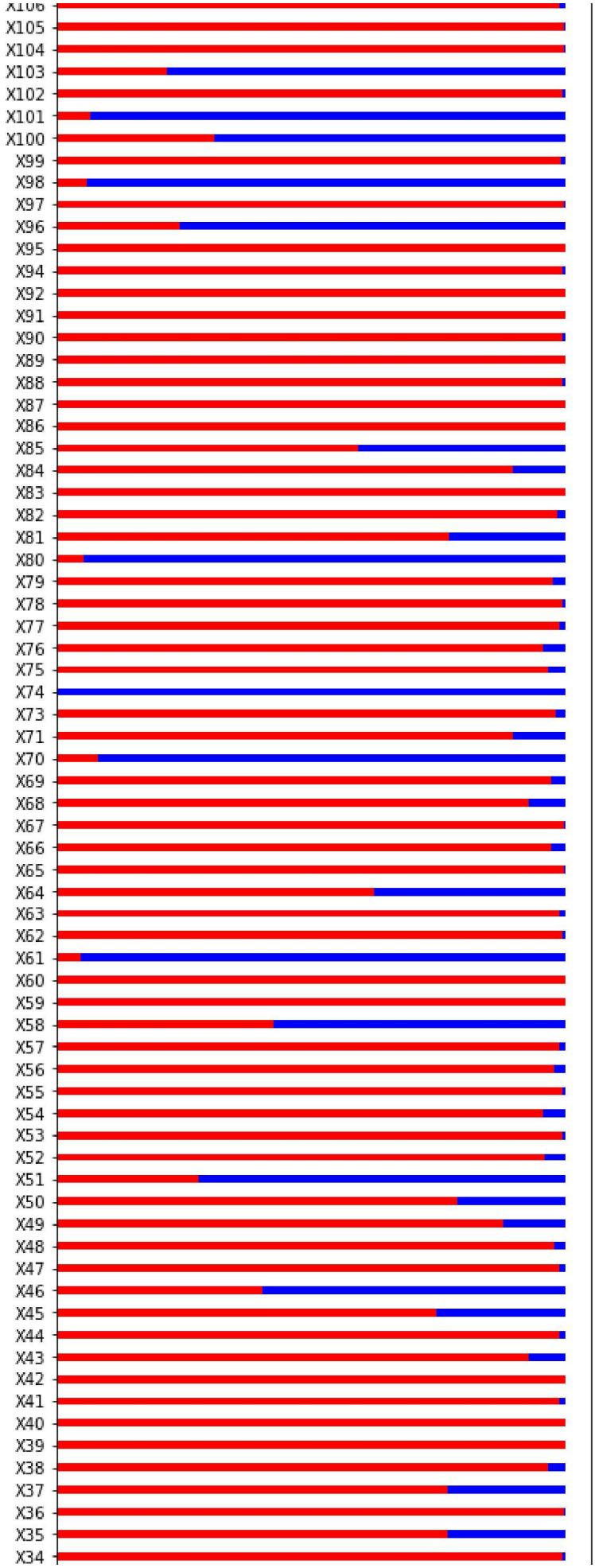
Binary Variables


Zero count
One Count
X385
X384
X383
X382
X380
X379
X378
X377
X376
X375
X374
X373
X372
X371
X370
X369
X368
X367
X366
X365
X364
X363
X362
X361
X360
X359
X358
X357
X356
X355
X354
X353
X352
X351
X350
X349
X348
X346
X345
X344
X343
X342
X341
X340
X339
X338
X337
X336
X335
X334
X333
X332


X331
X329
X328
X327
X326
X325
X324
X323
X322
X321
X320
X319
X318
X317
X316
X315
X314
X313
X312
X311
X310
X309
X308
X307
X306
X305
X304
X302
X301
X300
X299
X298
X296
X295
X294
X292
X291
X288
X287
X286
X285
X284
X283
X282
X281
X280
X279
X278
X277
X276
X275
X274
X273
X272
X271
X270
X269
X267
X266
X265
X264
X263
X262
X261
X260
X259
X258
X257
X256
X255


X253
X252
X251
X250
X249
X248
X247
X246
X245
X244
X243
X242
X241
X240
X239
X238
X237
X236
X234
X232
X231
X230
X229
X228
X227
X226
X225
X224
X223
X222
X221
X220
X219
X218
X217
X216
X215
X214
X213
X212
X211
X210
X209
X208
X207
X206
X205
X204
X203
X202
X201
X200
X199
X198
X197
X196
X195
X194
X192
X191
X190
X189
X187
X186
X185
X184
X183
X182
X181
X180


X179
X178
X177
X176
X175
X174
X173
X172
X171
X170
X169
X168
X167
X166
X165
X164
X163
X162
X161
X160
X159
X158
X157
X156
X155
X154
X153
X152
X151
X150
X148
X147
X146
X145
X144
X143
X142
X141
X140
X139
X138
X137
X136
X135
X134
X133
X132
X131
X130
X129
X128
X127
X126
X125
X124
X123
X122
X120
X119
X118
X117
X116
X115
X114
X113
X112
X111
X110
X109
X108


X106
X105
X104
X103
X102
X101
X100
X99
X98
X97
X96
X95
X94
X92
X91
X90
X89
X88
X87
X86
X85
X84
X83
X82
X81
X80
X79
X78
X77
X76
X75
X74
X73
X71
X70
X69
X68
X67
X66
X65
X64
X63
X62
X61
X60
X59
X58
X57
X56
X55
X54
X53
X52
X51
X50
X49
X48
X47
X46
X45
X44
X43
X42
X41
X40
X39
X38
X37
X36
X35
X34

The mean y value in each of the binary variable

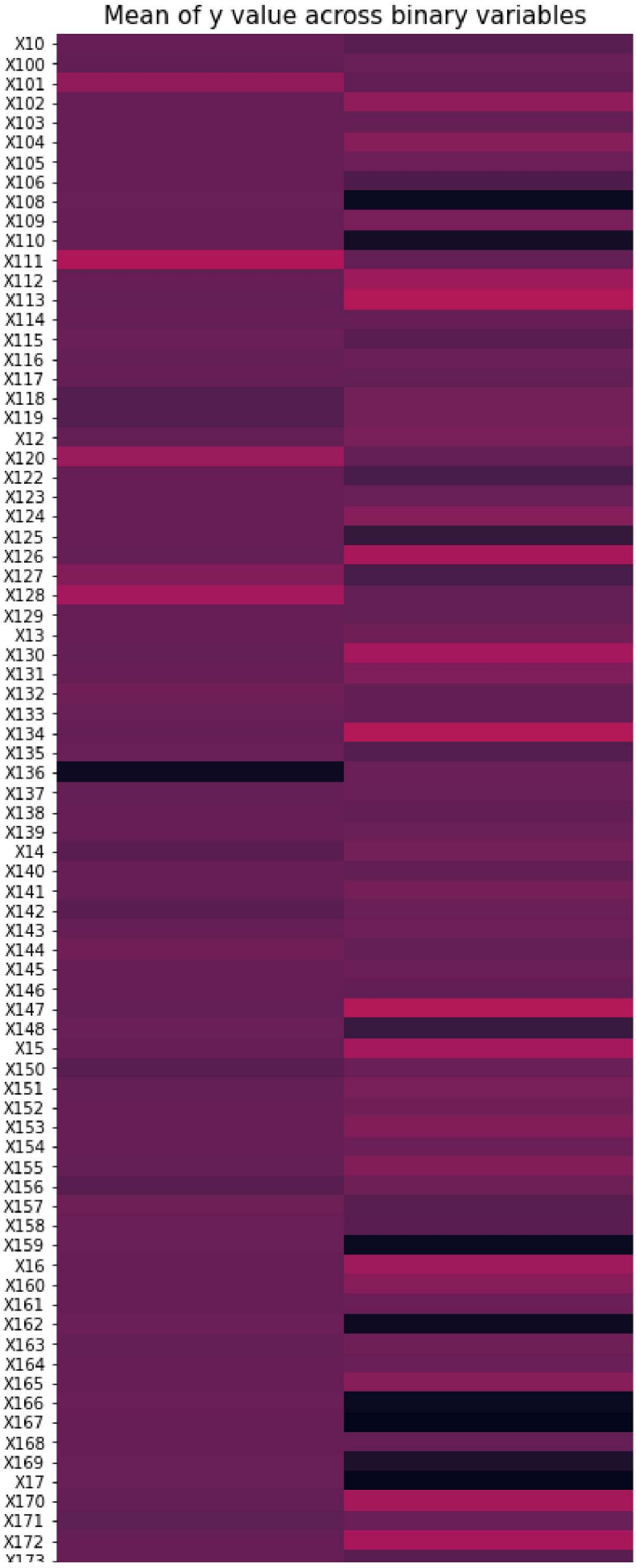

```
zero_mean_list = []
one_mean_list = []
cols_list = unique_values_dict['[0, 1]']
for col in cols_list:
    zero_mean_list.append(train_df.loc[train_df[col]==0].y.mean())
    one_mean_list.append(train_df.loc[train_df[col]==1].y.mean())
```

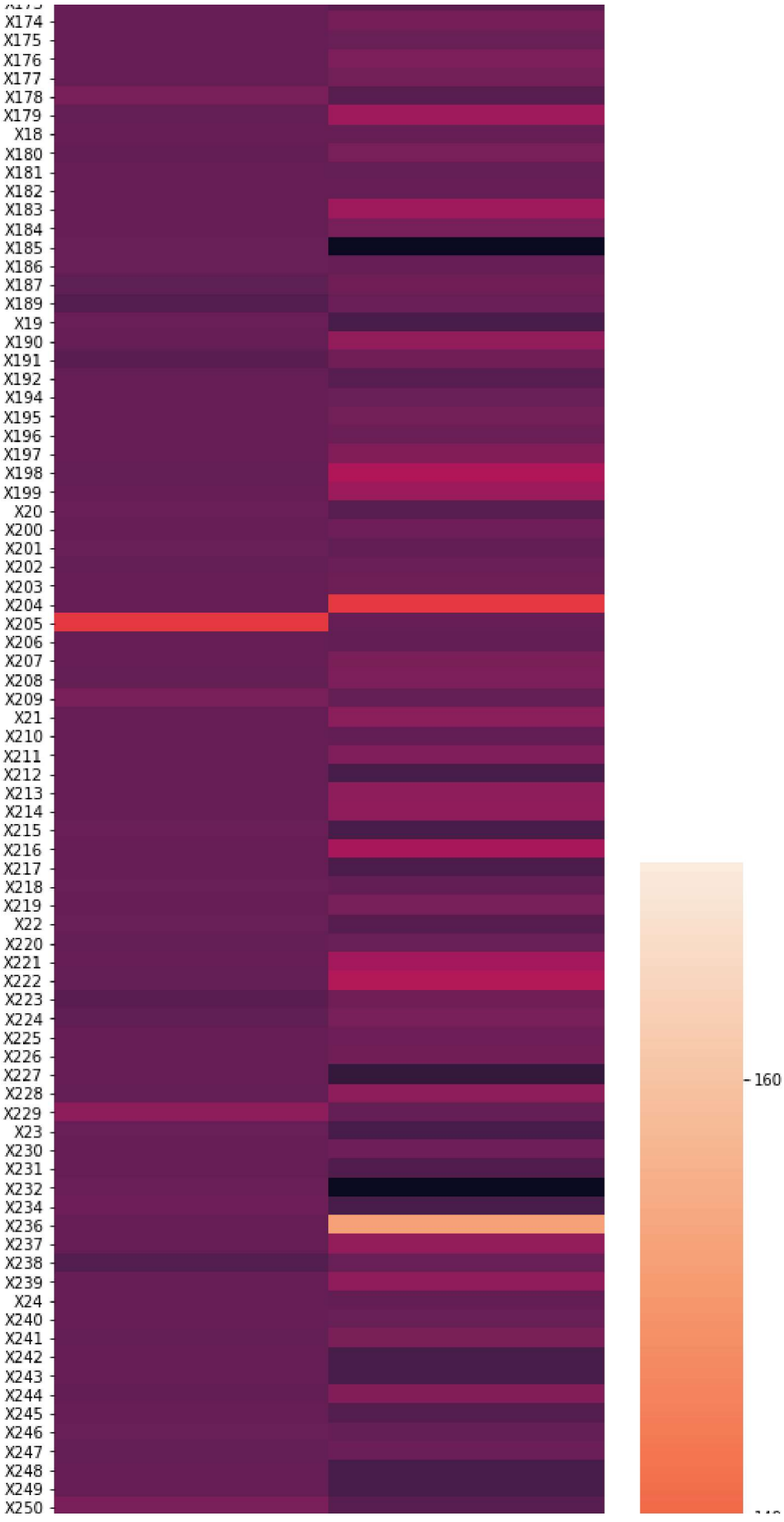

```
new_df = pd.DataFrame({"column_name":cols_list+cols_list, "value":[0]*len(cols_list) + [1]*le
                       "y_mean":zero_mean_list+one_mean_list})
new_df = new_df.pivot(index='column_name', columns='value', values='y_mean')
new_df
```

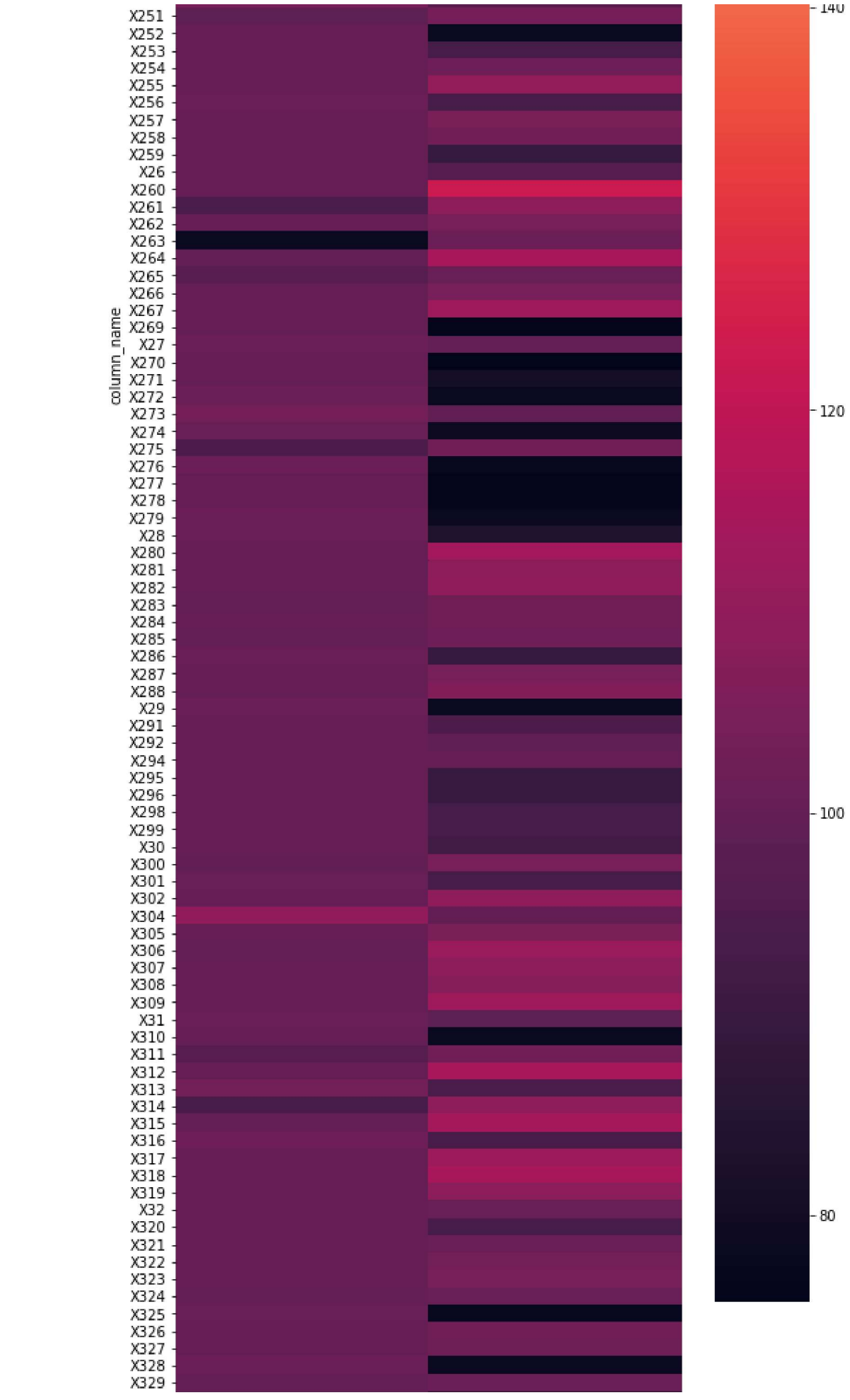
column_name	0	1
X10	100.688500	97.723214
X100	99.763083	101.046740
X101	110.198376	99.991894
X102	100.586823	109.617931
X103	100.472172	100.697632
...	...	...
X95	100.644223	120.950000
X96	97.968792	101.504108
X97	100.676829	94.180556
X98	110.267934	100.062264
X99	100.709446	93.647778

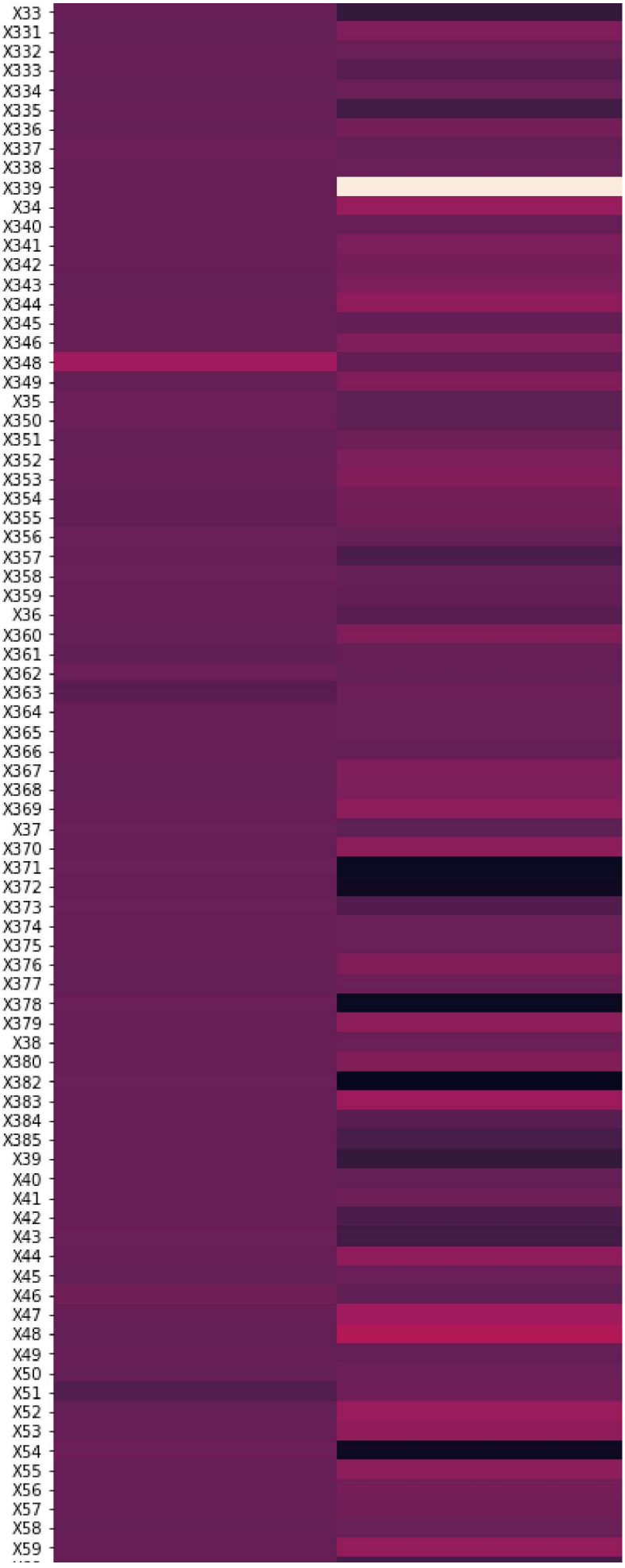
356 rows × 2 columns

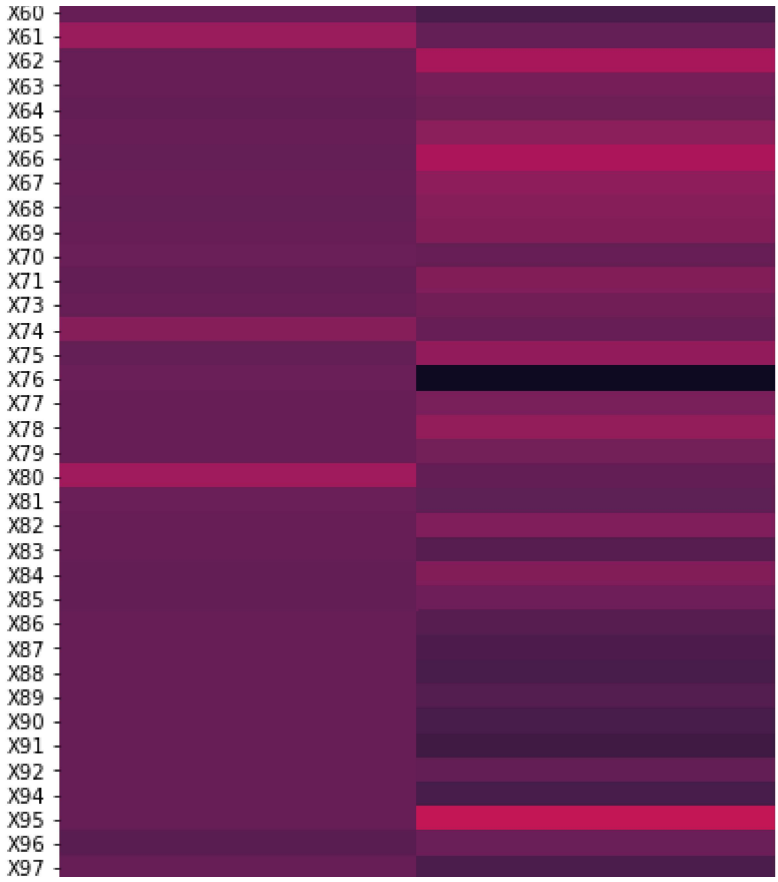
```
plt.figure(figsize=(8,80))
sns.heatmap(new_df)
plt.title("Mean of y value across binary variables", fontsize=15)
plt.show()
```


Mean of y value across binary variables
X10
X100
X101
X102
X103
X104
X105
X106
X108
X109
X110
X111
X112
X113
X114
X115
X116
X117
X118
X119
X12
X120
X122
X123
X124
X125
X126
X127
X128
X129
X13
X130
X131
X132
X133
X134
X135
X136
X137
X138
X139
X14
X140
X141
X142
X143
X144
X145
X146
X147
X148
X15
X150
X151
X152
X153
X154
X155
X156
X157
X158
X159
X16
X160
X161
X162
X163
X164
X165
X166
X167
X168
X169
X17
X170
X171
X172


X174
X175
X176
X177
X178
X179
X18
X180
X181
X182
X183
X184
X185
X186
X187
X189
X19
X190
X191
X192
X194
X195
X196
X197
X198
X199
X20
X200
X201
X202
X203
X204
X205
X206
X207
X208
X209
X21
X210
X211
X212
X213
X214
X215
X216
X217
X218
X219
X22
X220
X221
X222
X223
X224
X225
X226
X227
X228
X229
X23
X230
X231
X232
X234
X236
X237
X238
X239
X24
X240
X241
X242
X243
X244
X245
X246
X247
X248
X249
X250
160


column_name
X251
X252
X253
X254
X255
X256
X257
X258
X259
X26
X260
X261
X262
X263
X264
X265
X266
X267
X269
X27
X270
X271
X272
X273
X274
X275
X276
X277
X278
X279
X28
X280
X281
X282
X283
X284
X285
X286
X287
X288
X29
X291
X292
X294
X295
X296
X298
X299
X30
X300
X301
X302
X304
X305
X306
X307
X308
X309
X31
X310
X311
X312
X313
X314
X315
X316
X317
X318
X319
X32
X320
X321
X322
X323
X324
X325
X326
X327
X328
X329
140
120
100
80


X33
X331
X332
X333
X334
X335
X336
X337
X338
X339
X34
X340
X341
X342
X343
X344
X345
X346
X348
X349
X35
X350
X351
X352
X353
X354
X355
X356
X357
X358
X359
X36
X360
X361
X362
X363
X364
X365
X366
X367
X368
X369
X37
X370
X371
X372
X373
X374
X375
X376
X377
X378
X379
X38
X380
X382
X383
X384
X385
X39
X40
X41
X42
X43
X44
X45
X46
X47
X48
X49
X50
X51
X52
X53
X54
X55
X56
X57
X58
X59

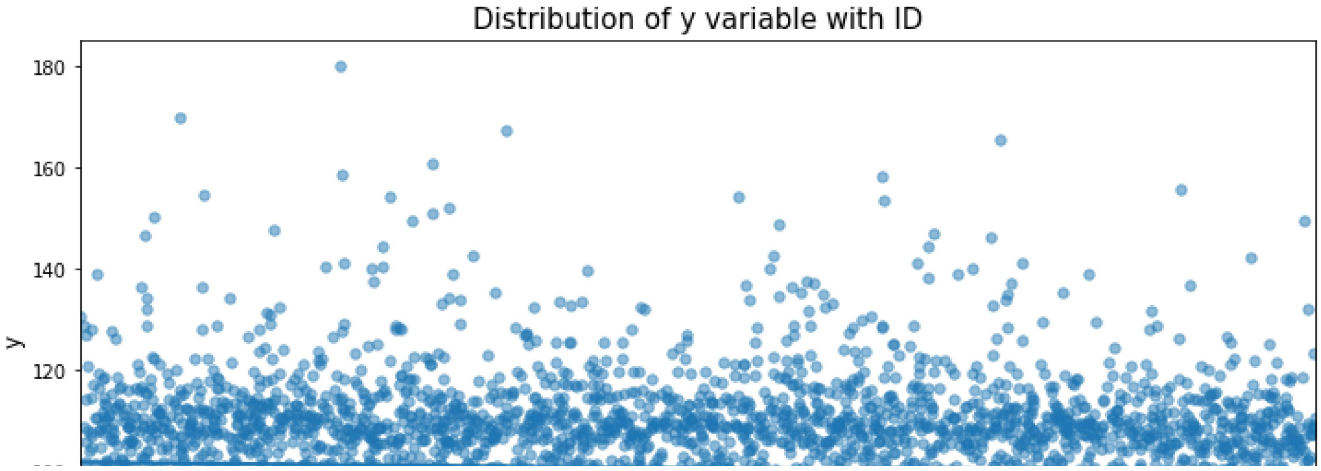


ID variable

One more important thing we need to look at it is ID variable. This will give an idea of how the splits are done across train and test (random or id based) and also to help see if ID has some potential prediction capability (probably not so useful for business)

Let us first see how the 'y' variable changes with ID variable.

```
var_name = "ID"
plt.figure(figsize=(12,6))
sns.regplot(x=var_name, y='y', data=train_df, scatter_kws={'alpha':0.5, 's':30})
plt.xlabel(var_name, fontsize=12)
plt.ylabel('y', fontsize=12)
plt.title("Distribution of y variable with "+var_name, fontsize=15)
plt.show()
```



There seems to be a slight decreasing trend with respect to ID variable. Now let us see how the IDs are distributed across train and test.

```
plt.figure(figsize=(6,10))
train_df['eval_set'] = "train"
test_df['eval_set'] = "test"
full_df = pd.concat([train_df[["ID","eval_set"]], test_df[["ID","eval_set"]]], axis=0)
full_df
```

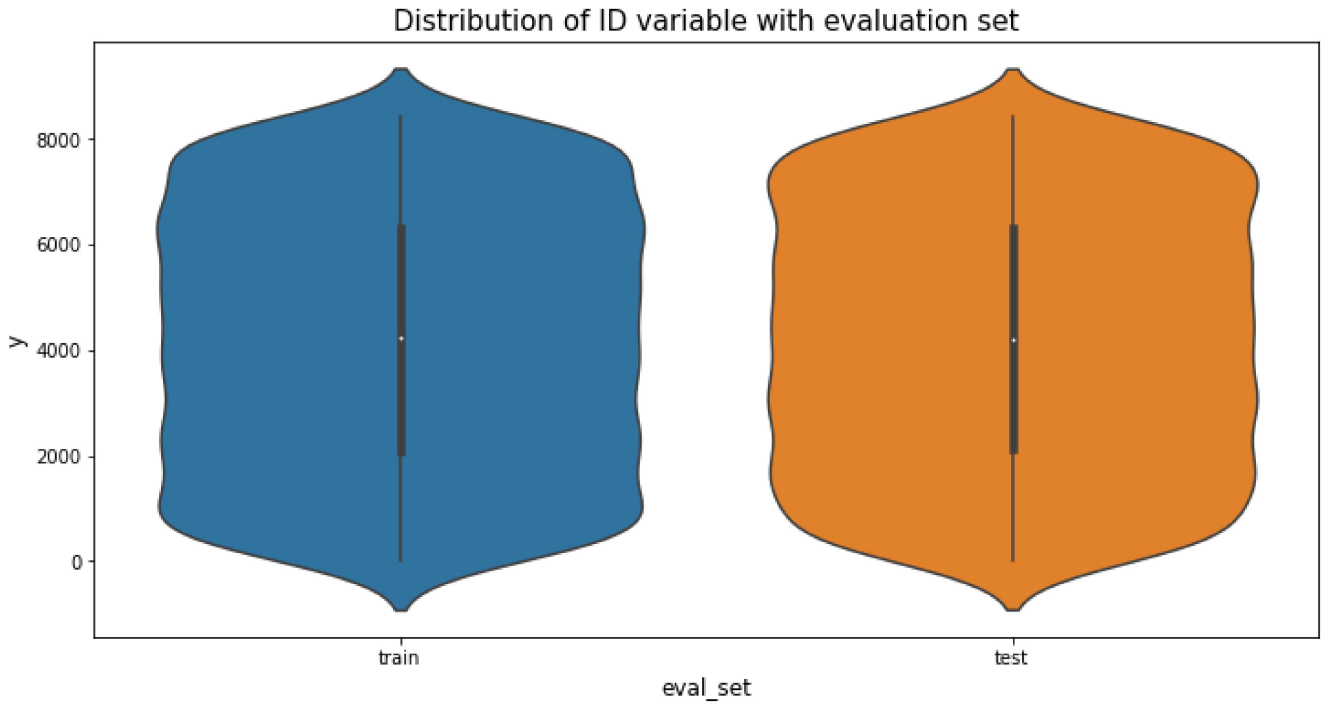
	ID	eval_set
0	0	train
1	6	train
2	7	train
3	9	train
4	13	train
...	...	...
4204	8410	test
4205	8411	test
4206	8413	test
4207	8414	test
4208	8416	test

8418 rows × 2 columns

```
<Figure size 432x720 with 0 Axes>
```

```
plt.figure(figsize=(12,6))
sns.violinplot(x="eval_set", y='ID', data=full_df)
plt.xlabel("eval_set", fontsize=12)
plt.ylabel('y', fontsize=12)
```

```
plt.title("Distribution of ID variable with evaluation set", fontsize=15)
plt.show()
```



Seems like a random split of ID variable between train and test samples.

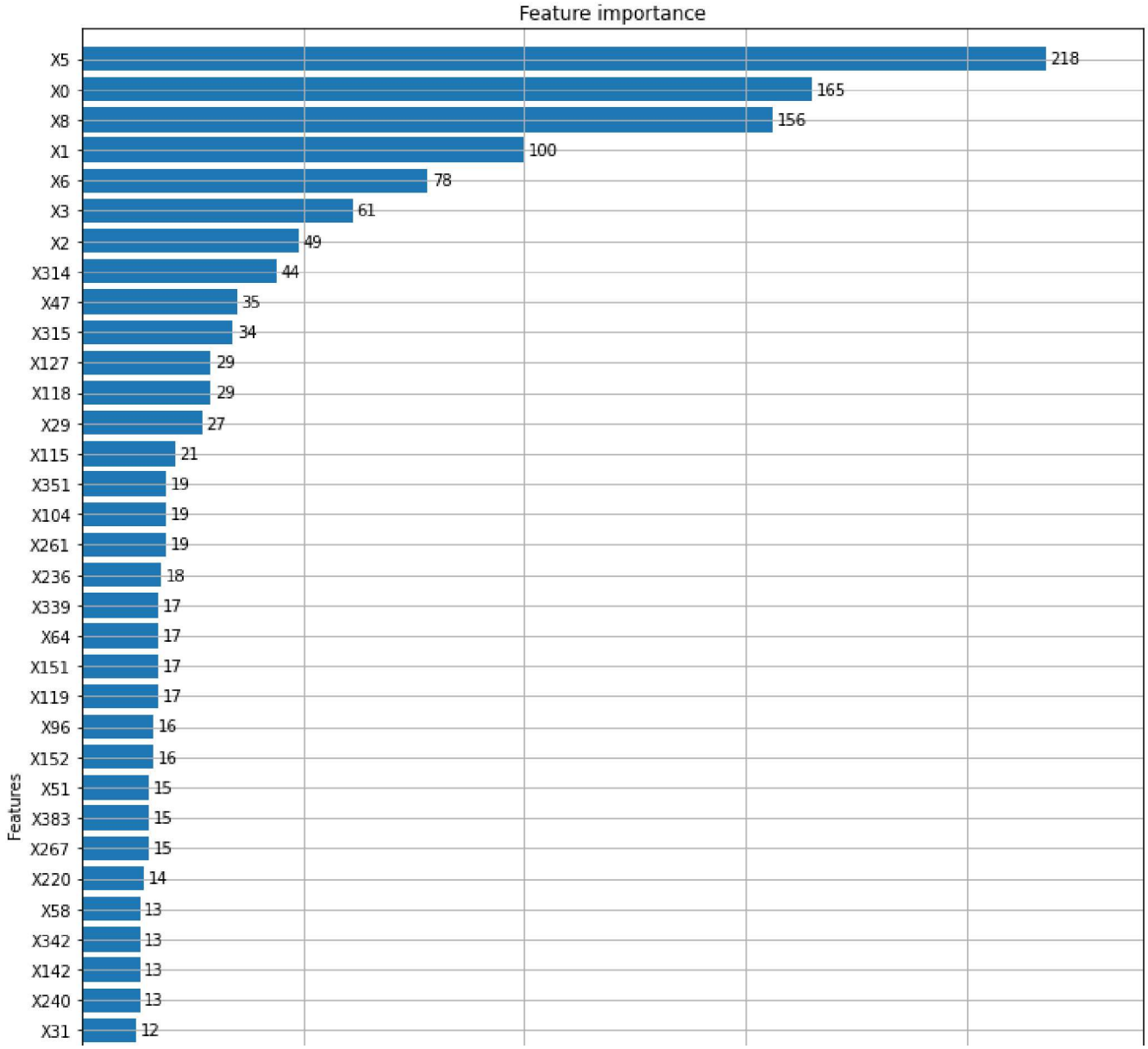
XGBoost

Important Variables:

Now let us run and xgboost model to get the important variables.

```
for f in ["X0", "X1", "X2", "X3", "X4", "X5", "X6", "X8"]:
        lbl = preprocessing.LabelEncoder()
        lbl.fit(list(train_df[f].values))
        train_df[f] = lbl.transform(list(train_df[f].values))

train_y = train_df['y'].values
train_X = train_df.drop(["ID", "y", "eval_set"], axis=1)


def xgb_r2_score(preds, dtrain):
    labels = dtrain.get_label()
    return 'r2', r2_score(labels, preds)


for f in ["X0", "X1", "X2", "X3", "X4", "X5", "X6", "X8"]:
        lbl = preprocessing.LabelEncoder()
```

```
[10:04:29] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now d
```

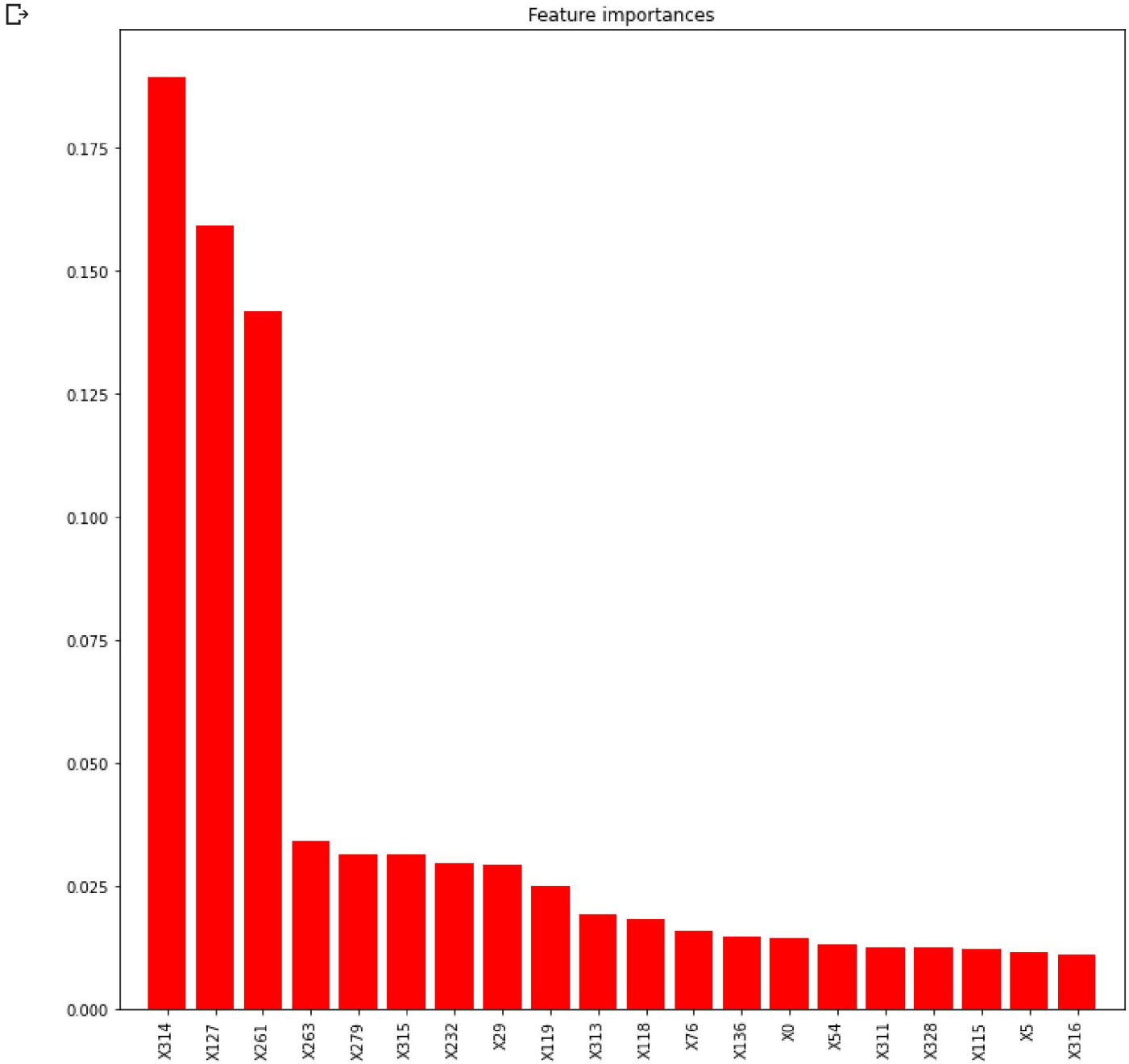


Categorical occupy the top spots followed by binary variables.

Let us also build a Random Forest model and check the important variables.


```
from sklearn.ensemble import RandomForestRegressor
model = RandomForestRegressor(n_estimators=200, max_depth=10, min_samples_leaf=4, max_feature
model.fit(train_X, train_y)
feat_names = train_X.columns.values

## plot the importances ##
importances = model.feature_importances_
std = np.std([tree.feature_importances_ for tree in model.estimators_], axis=0)
indices = np.argsort(importances)[::-1][:20]

plt.figure(figsize=(12,12))
plt.title("Feature importances")
plt.bar(range(len(indices)), importances[indices], color="r", align="center")
```

```
plt.xticks(range(len(indices)), feat_names[indices], rotation='vertical')
plt.xlim([-1, len(indices)])
plt.show()
```



Quite a few differences in the important variables between xgboost and random forest. Not sure why though.!