Reduce the time a Mercedes-Benz spends on the test bench.

Problem Statement Scenario: Since the first automobile, the Benz Patent Motor Car in 1886, Mercedes-Benz has stood for important automotive innovations. These include the passenger safety cell with a crumple zone, the airbag, and intelligent assistance systems. Mercedes-Benz applies for nearly 2000 patents per year, making the brand the European leader among premium carmakers. Mercedes-Benz is the leader in the premium car industry. With a huge selection of features and options, customers can choose the customized Mercedes-Benz of their dreams.

To ensure the safety and reliability of every unique car configuration before they hit the road, the company's engineers have developed a robust testing system. As one of the world's biggest manufacturers of premium cars, safety and efficiency are paramount on Mercedes-Benz's production lines. However, optimizing the speed of their testing system for many possible feature combinations is complex and time-consuming without a powerful algorithmic approach.

You are required to reduce the time that cars spend on the test bench. Others will work with a dataset representing different permutations of features in a Mercedes-Benz car to predict the time it takes to pass testing. Optimal algorithms will contribute to faster testing, resulting in lower carbon dioxide emissions without reducing Mercedes-Benz's standards.

Following actions should be performed:

If for any column(s), the variance is equal to zero, then you need to remove those variable(s). Check for null and unique values for test and train sets. Apply label encoder. Perform dimensionality reduction. Predict your test_df values using XGBoost.

In [ ]:
```
#Let us first import the necessary modules.
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
```

In [35]:
```
df_train = pd.read_csv("train.csv")
df_test = pd.read_csv("test.csv")
print("Train shape : ", df_train.shape)
print("Test shape : ", df_test.shape)
```

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

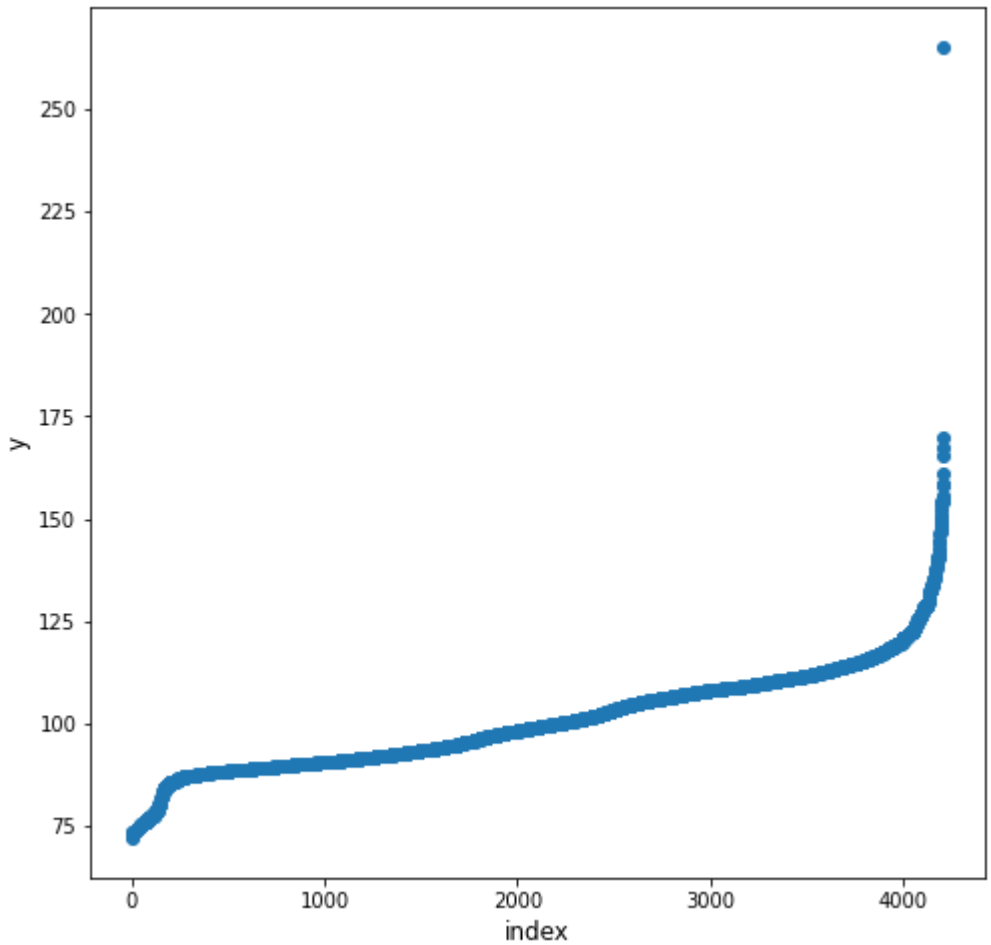
In [36]:
```
# Let us look at the top few rows.
df_train.head()
```

Out[36]:

	ID	y	X0	X1	X2	X3	X4	X5	X6	X8	X10	X11	X12	X13	X14	X15	X16	X17	X18	X1
0	0	130.81	k	v	at	a	d	u	j	o	0	0	0	1	0	0	0	0	1	
1	6	88.53	k	t	av	e	d	y	l	o	0	0	0	0	0	0	0	0	1	
2	7	76.26	az	w	n	c	d	x	j	x	0	0	0	0	0	0	0	1	0	
3	9	80.62	az	t	n	f	d	x	l	e	0	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	0	

In [37]:

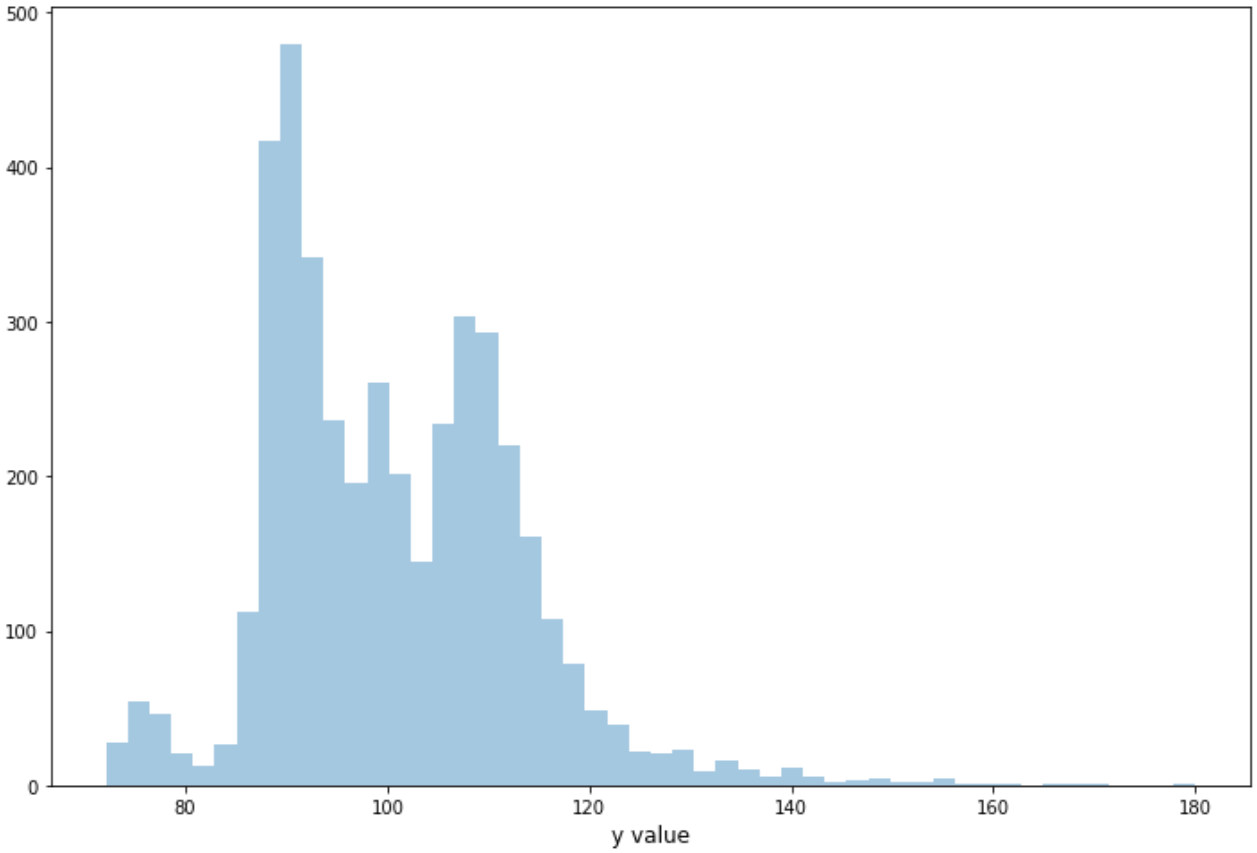
```
plt.figure(figsize=(8,8))
plt.scatter(range(df_train.shape[0]),np.sort(df_train.y.values))
plt.xlabel('index', fontsize=12)
plt.ylabel('y', fontsize=12)
plt.show()
```



In [38]:

```
ulimit = 180
df_train['y'].iloc[df_train['y']>ulimit] = ulimit

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

```
C:\Anaconda\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` i
s a deprecated function and will be removed in a future version. Please adapt your code
to use either `displot` (a figure-level function with similar flexibility) or `histplot`
(an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [39]:
```
# Now let us have a look at the data type of all the variables present in the dataset
dtype_data=df_train.dtypes.reset_index()
dtype_data.columns = ["Count", "Column Type"]
dtype_data.groupby("Column Type").aggregate('count').reset_index()
```

Out[39]:

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

In [40]:
```
dtype_data.loc[:10,:]
```

Out[40]:

	Count	Column Type
0	ID	int64
1	y	float64
2	X0	object
3	X1	object

	Count	Column Type
4	X2	object
5	X3	object
6	X4	object
7	X5	object
8	X6	object
9	X8	object
10	X10	int64

In [41]:
```
df_train.isnull().sum().sum()
```

Out[41]:
```
0
```

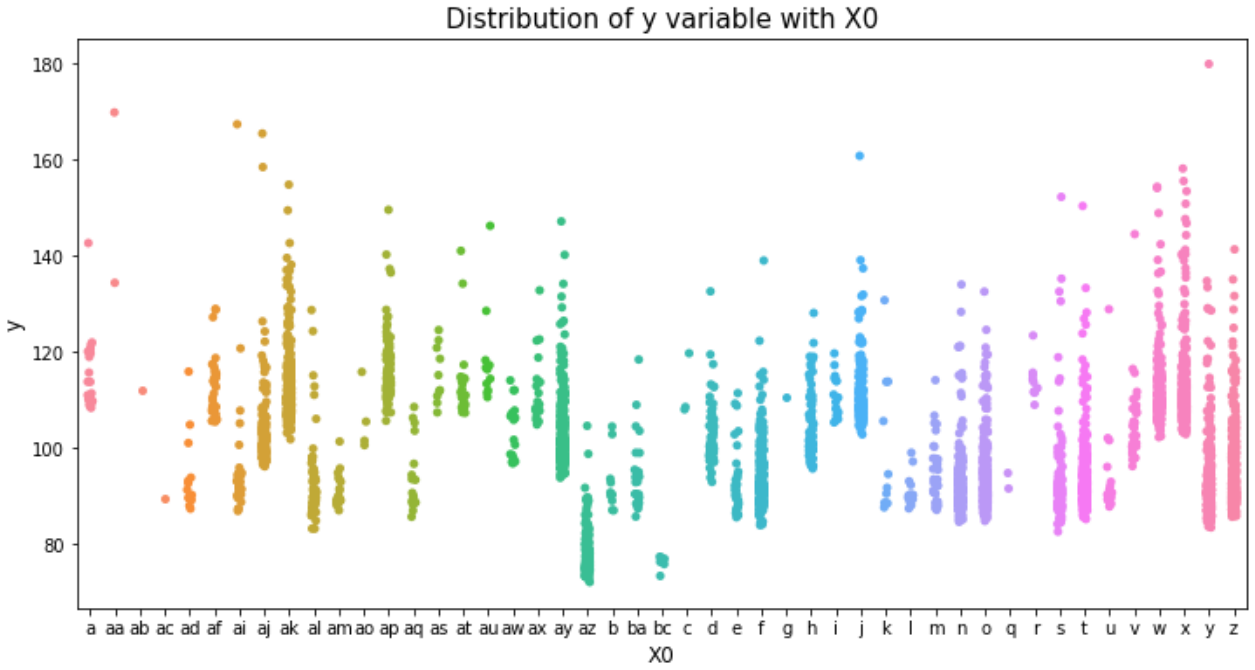
In [42]:
```
# Integer Columns Analysis
unique_value_dict = {}
for col in df_train.columns:
    if col not in ["ID", "y", "X0", "X1", "X2", "X3", "X4", "X5", "X6", "X8"]:
        unique_value = str(np.sort(df_train[col].unique()).tolist())
        t_list = unique_value_dict.get(unique_value, [])
        t_list.append(col)
        unique_value_dict[unique_value] = t_list[:]
for unique_val, columns in unique_value_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', 'X2
3', 'X24', 'X26', 'X27', 'X28', 'X29', 'X30', 'X31', 'X32', 'X33', 'X34', 'X35', 'X36',
'X37', 'X38', 'X39', 'X40', 'X41', 'X42', 'X43', 'X44', 'X45', 'X46', 'X47', 'X48', 'X4
9', 'X50', 'X51', 'X52', 'X53', 'X54', 'X55', 'X56', 'X57', 'X58', 'X59', 'X60', 'X61',
'X62', 'X63', 'X64', 'X65', 'X66', 'X67', 'X68', 'X69', 'X70', 'X71', 'X73', 'X74', 'X7
5', 'X76', 'X77', 'X78', 'X79', 'X80', 'X81', 'X82', 'X83', 'X84', 'X85', 'X86', 'X87',
'X88', 'X89', 'X90', 'X91', 'X92', 'X94', 'X95', 'X96', 'X97', 'X98', 'X99', 'X100', 'X1
01', 'X102', 'X103', 'X104', 'X105', 'X106', 'X108', 'X109', 'X110', 'X111', 'X112', 'X1
13', 'X114', 'X115', 'X116', 'X117', 'X118', 'X119', 'X120', 'X122', 'X123', 'X124', 'X1
25', 'X126', 'X127', 'X128', 'X129', 'X130', 'X131', 'X132', 'X133', 'X134', 'X135', 'X1
36', 'X137', 'X138', 'X139', 'X140', 'X141', 'X142', 'X143', 'X144', 'X145', 'X146', 'X1
47', 'X148', 'X150', 'X151', 'X152', 'X153', 'X154', 'X155', 'X156', 'X157', 'X158', 'X1
59', 'X160', 'X161', 'X162', 'X163', 'X164', 'X165', 'X166', 'X167', 'X168', 'X169', 'X1
70', 'X171', 'X172', 'X173', 'X174', 'X175', 'X176', 'X177', 'X178', 'X179', 'X180', 'X1
81', 'X182', 'X183', 'X184', 'X185', 'X186', 'X187', 'X189', 'X190', 'X191', 'X192', 'X1
94', 'X195', 'X196', 'X197', 'X198', 'X199', 'X200', 'X201', 'X202', 'X203', 'X204', 'X2
05', 'X206', 'X207', 'X208', 'X209', 'X210', 'X211', 'X212', 'X213', 'X214', 'X215', 'X2
16', 'X217', 'X218', 'X219', 'X220', 'X221', 'X222', 'X223', 'X224', 'X225', 'X226', 'X2
27', 'X228', 'X229', 'X230', 'X231', 'X232', 'X234', 'X236', 'X237', 'X238', 'X239', 'X2
40', 'X241', 'X242', 'X243', 'X244', 'X245', 'X246', 'X247', 'X248', 'X249', 'X250', 'X2
51', 'X252', 'X253', 'X254', 'X255', 'X256', 'X257', 'X258', 'X259', 'X260', 'X261', 'X2
62', 'X263', 'X264', 'X265', 'X266', 'X267', 'X269', 'X270', 'X271', 'X272', 'X273', 'X2
74', 'X275', 'X276', 'X277', 'X278', 'X279', 'X280', 'X281', 'X282', 'X283', 'X284', 'X2
85', 'X286', 'X287', 'X288', 'X291', 'X292', 'X294', 'X295', 'X296', 'X298', 'X299', 'X3
00', 'X301', 'X302', 'X304', 'X305', 'X306', 'X307', 'X308', 'X309', 'X310', 'X311', 'X3
```

```
12', 'X313', 'X314', 'X315', 'X316', 'X317', 'X318', 'X319', 'X320', 'X321', 'X322', 'X3
23', 'X324', 'X325', 'X326', 'X327', 'X328', 'X329', 'X331', 'X332', 'X333', 'X334', 'X3
35', 'X336', 'X337', 'X338', 'X339', 'X340', 'X341', 'X342', 'X343', 'X344', 'X345', 'X3
46', 'X348', 'X349', 'X350', 'X351', 'X352', 'X353', 'X354', 'X355', 'X356', 'X357', 'X3
58', 'X359', 'X360', 'X361', 'X362', 'X363', 'X364', 'X365', 'X366', 'X367', 'X368', 'X3
69', 'X370', 'X371', 'X372', 'X373', 'X374', 'X375', 'X376', 'X377', 'X378', 'X379', 'X3
80', 'X382', 'X383', 'X384', 'X385']
----------------------------------------------------------
Columns containing the unique values :  [0]
['X11', 'X93', 'X107', 'X233', 'X235', 'X268', 'X289', 'X290', 'X293', 'X297', 'X330',
'X347']
----------------------------------------------------------
```

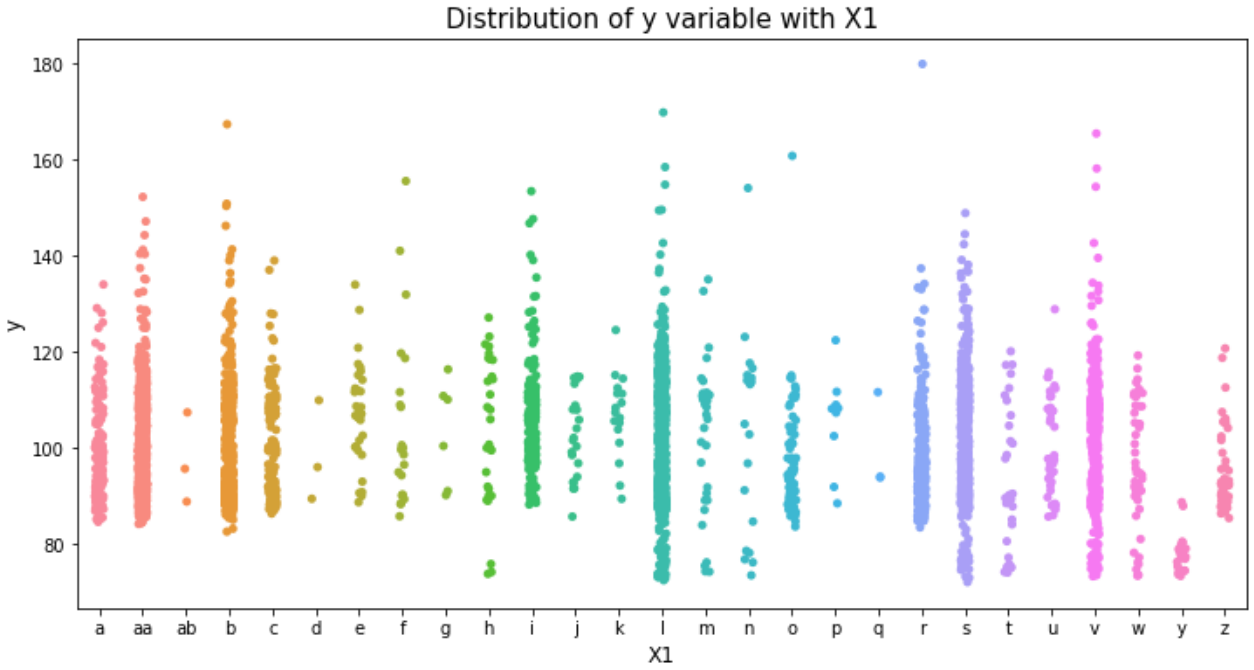
In [43]:

```
# Now let us explore the categorical columns present in the dataset.
var="X0"
colu_order=np.sort(df_train[var].unique()).tolist()
plt.figure(figsize=(12,6))
sns.stripplot(x=var,y="y",data=df_train,order=colu_order)
plt.xlabel(var,fontsize=12)
plt.ylabel("y",fontsize=12)
plt.title("Distribution of y variable with "+var, fontsize=15)
plt.show()
```



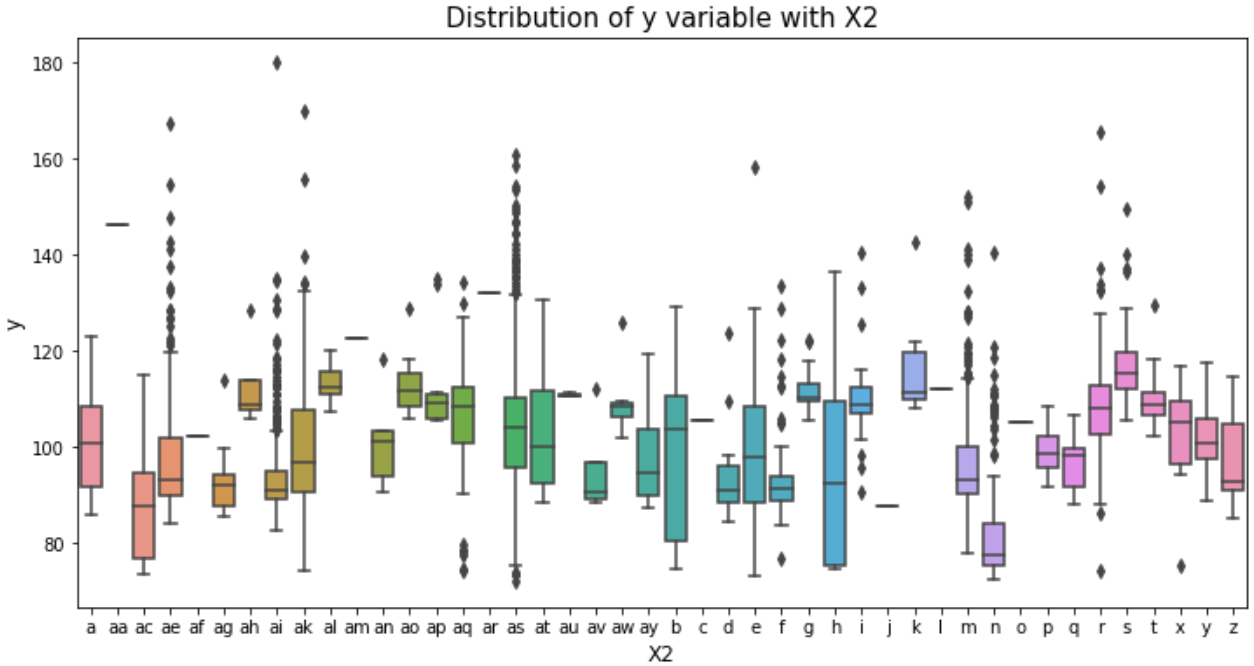
In [44]:

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



In [45]:

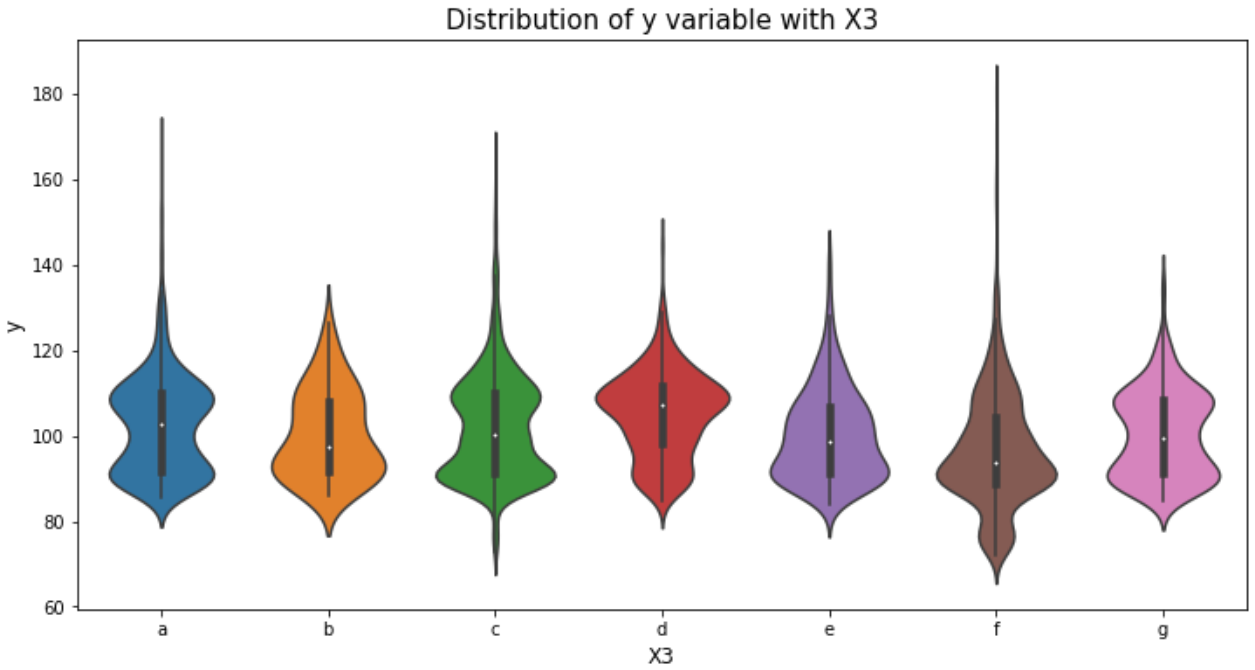
```
var="X2"
colu_order=np.sort(df_train[var].unique()).tolist()
plt.figure(figsize=(12,6))
sns.boxplot(x=var,y="y",data=df_train,order=colu_order)
plt.xlabel(var,fontsize=12)
plt.ylabel("y",fontsize=12)
plt.title("Distribution of y variable with "+var, fontsize=15)
plt.show()
```



In [46]:

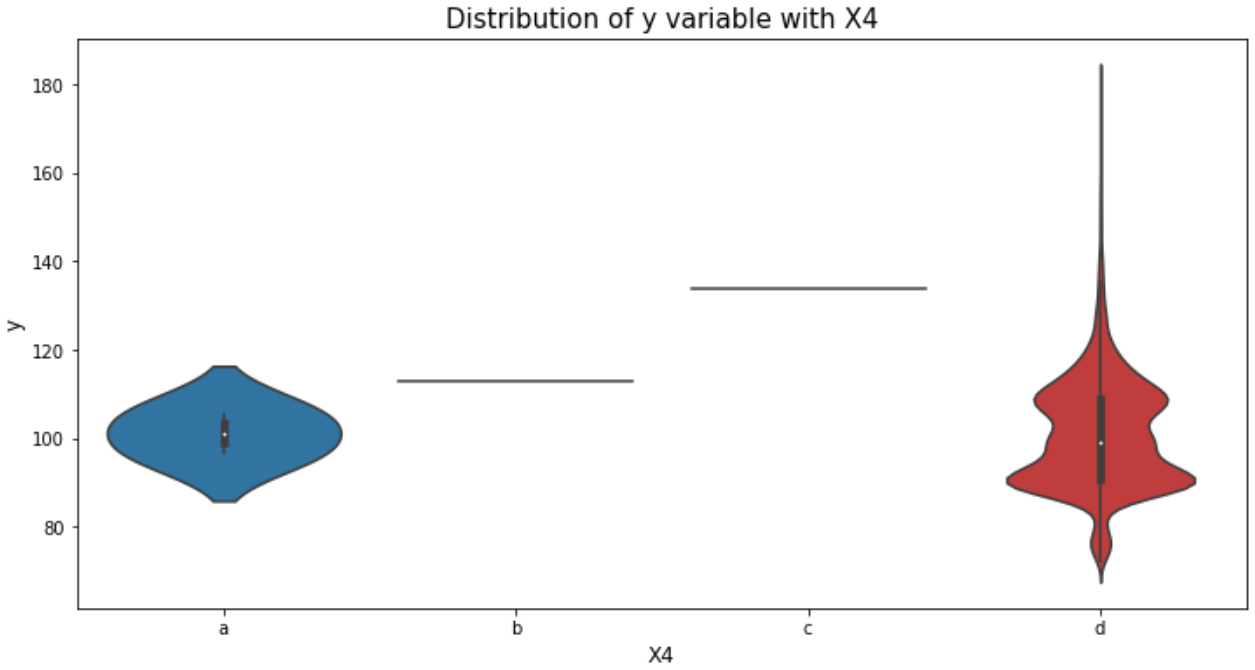
```
var="X3"
colu_order=np.sort(df_train[var].unique()).tolist()
plt.figure(figsize=(12,6))
sns.violinplot(x=var,y="y",data=df_train,order=colu_order)
plt.xlabel(var,fontsize=12)
```

```
plt.ylabel("y",fontsize=12)
plt.title("Distribution of y variable with "+var, fontsize=15)
plt.show()
```



In [47]:

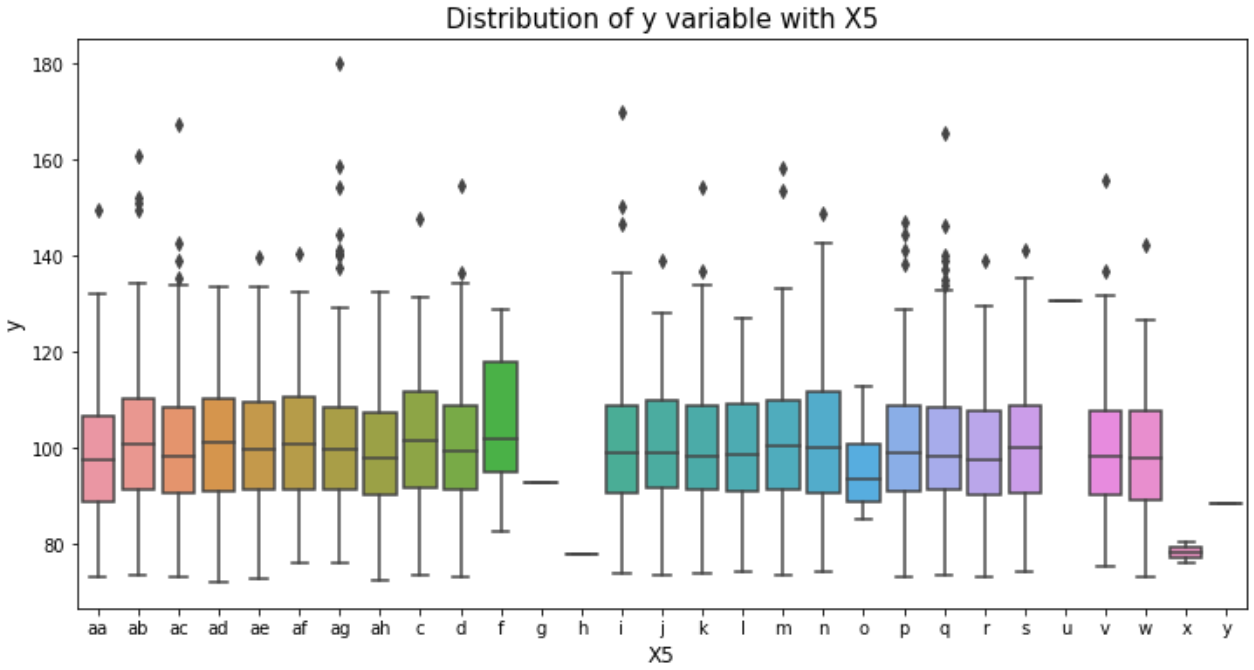
```
var="X4"
colu_order=np.sort(df_train[var].unique()).tolist()
plt.figure(figsize=(12,6))
sns.violinplot(x=var,y="y",data=df_train,order=colu_order)
plt.xlabel(var,fontsize=12)
plt.ylabel("y",fontsize=12)
plt.title("Distribution of y variable with "+var, fontsize=15)
plt.show()
```



In [48]:

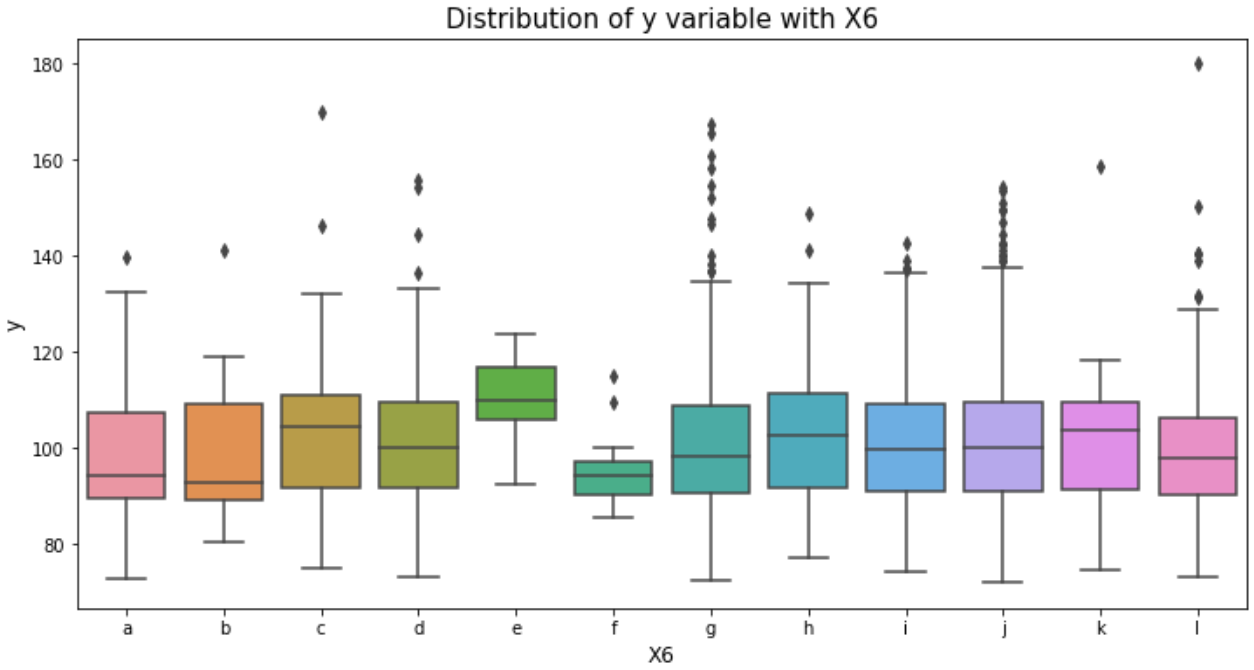
```
var="X5"
```

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



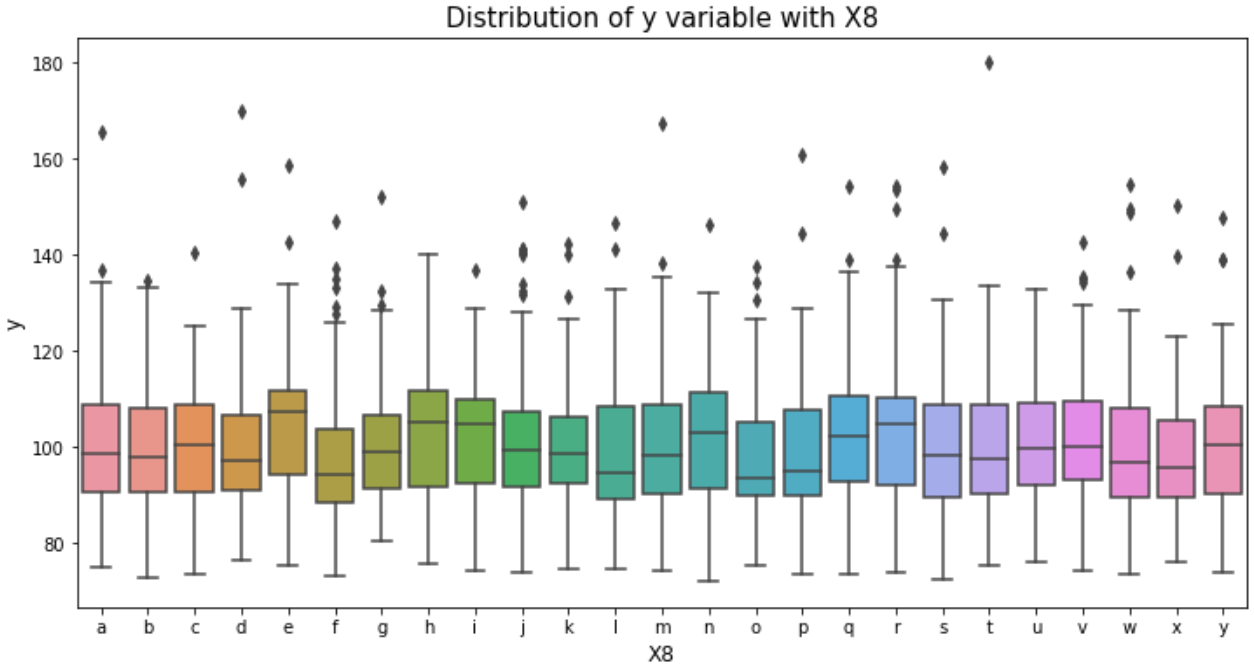
In [49]:

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



In [50]:

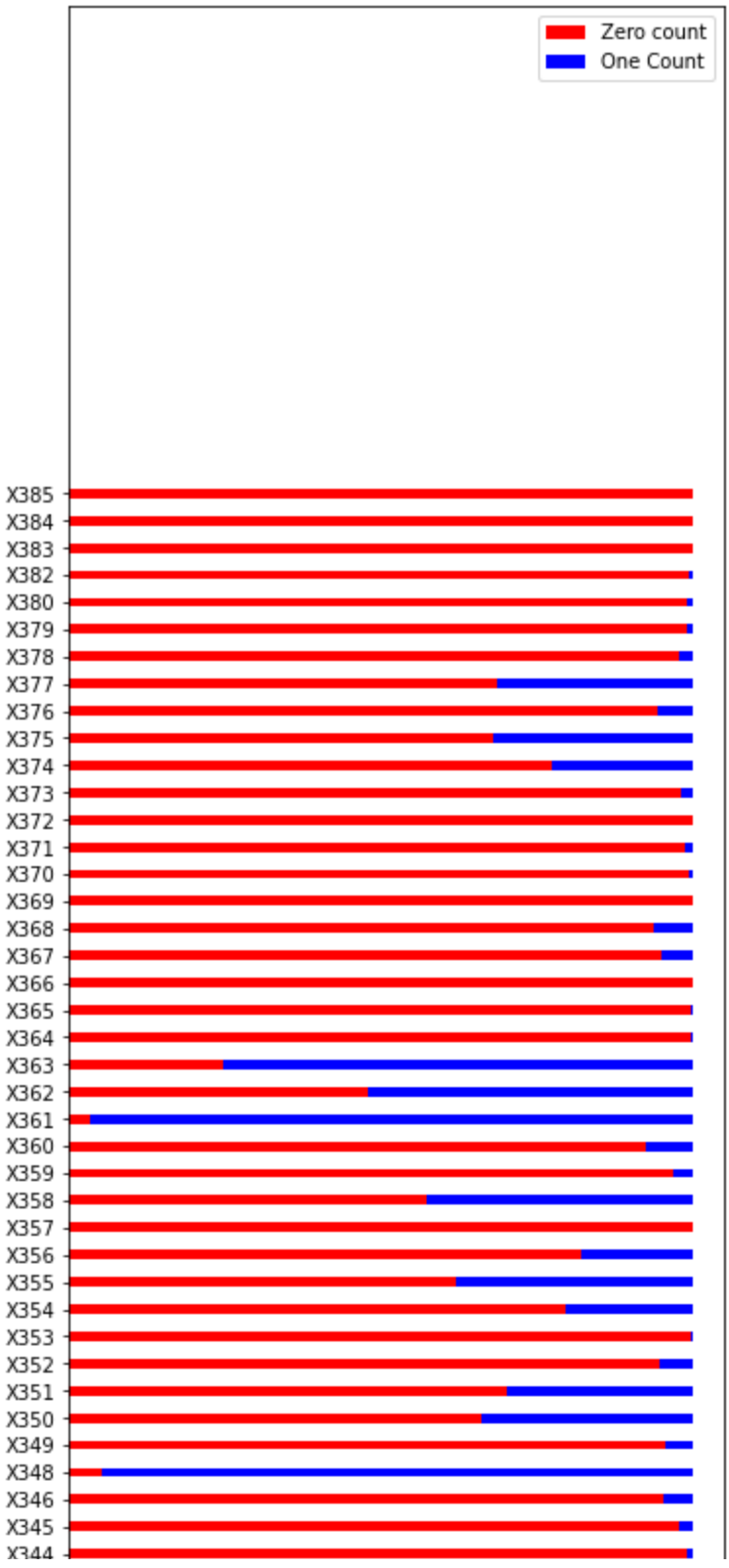
```
var="X8"
colu_order=np.sort(df_train[var].unique()).tolist()
plt.figure(figsize=(12,6))
sns.boxplot(x=var,y="y",data=df_train,order=colu_order)
plt.xlabel(var,fontsize=12)
plt.ylabel("y",fontsize=12)
plt.title("Distribution of y variable with "+var, fontsize=15)
plt.show()
```

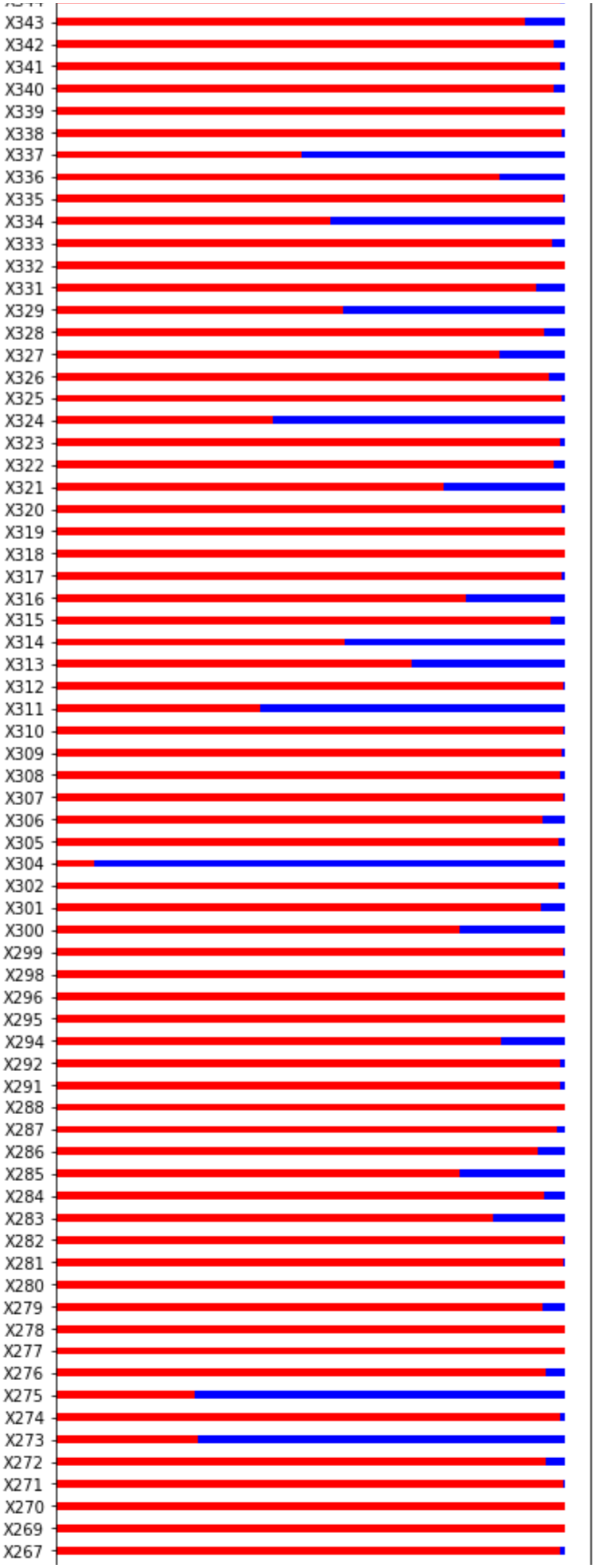


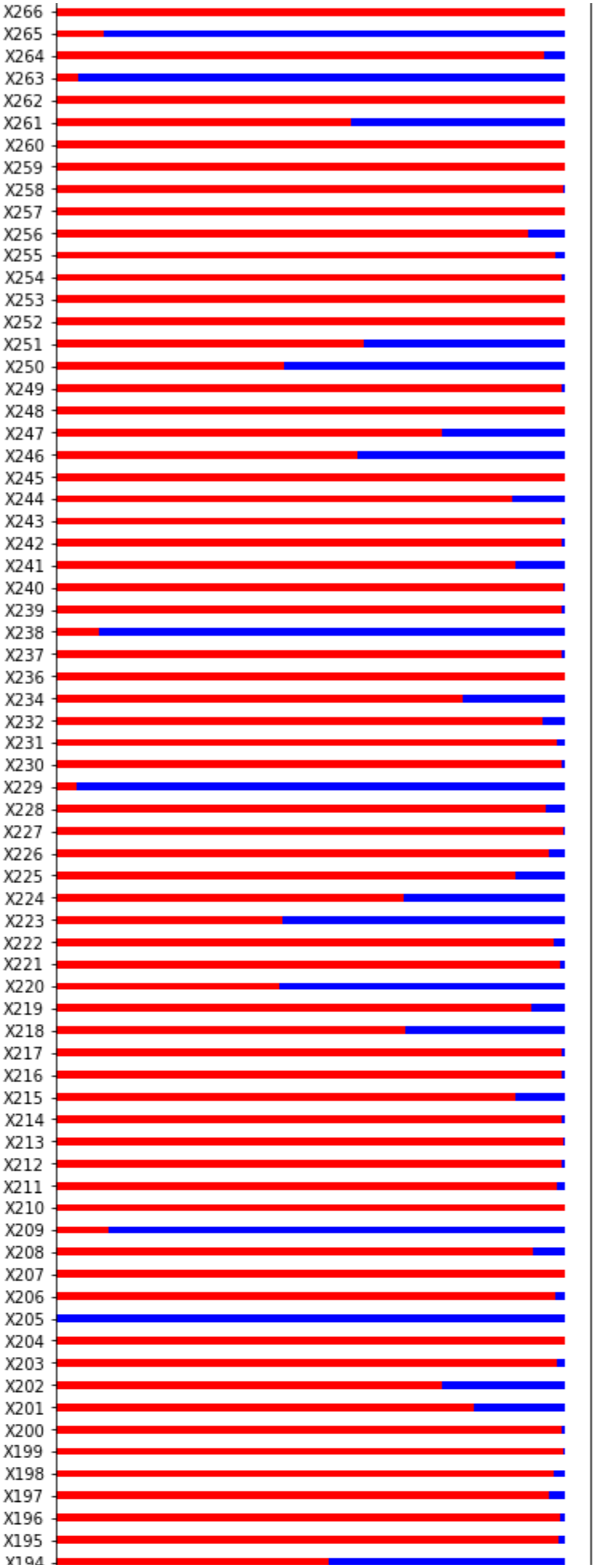
In [51]:

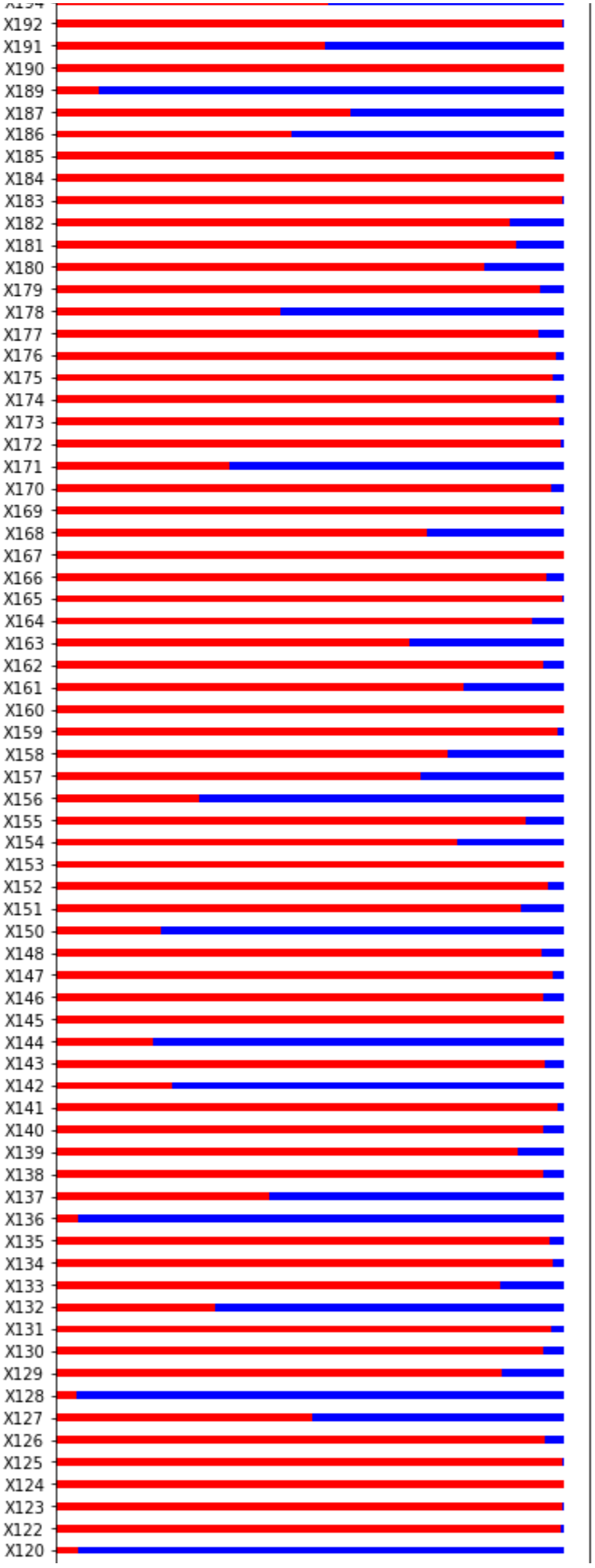
```
zero_list=[]
one_list=[]
col_list = unique_value_dict['[0, 1]']
for col in col_list:
    zero_list.append((df_train[col]==0).sum())
```

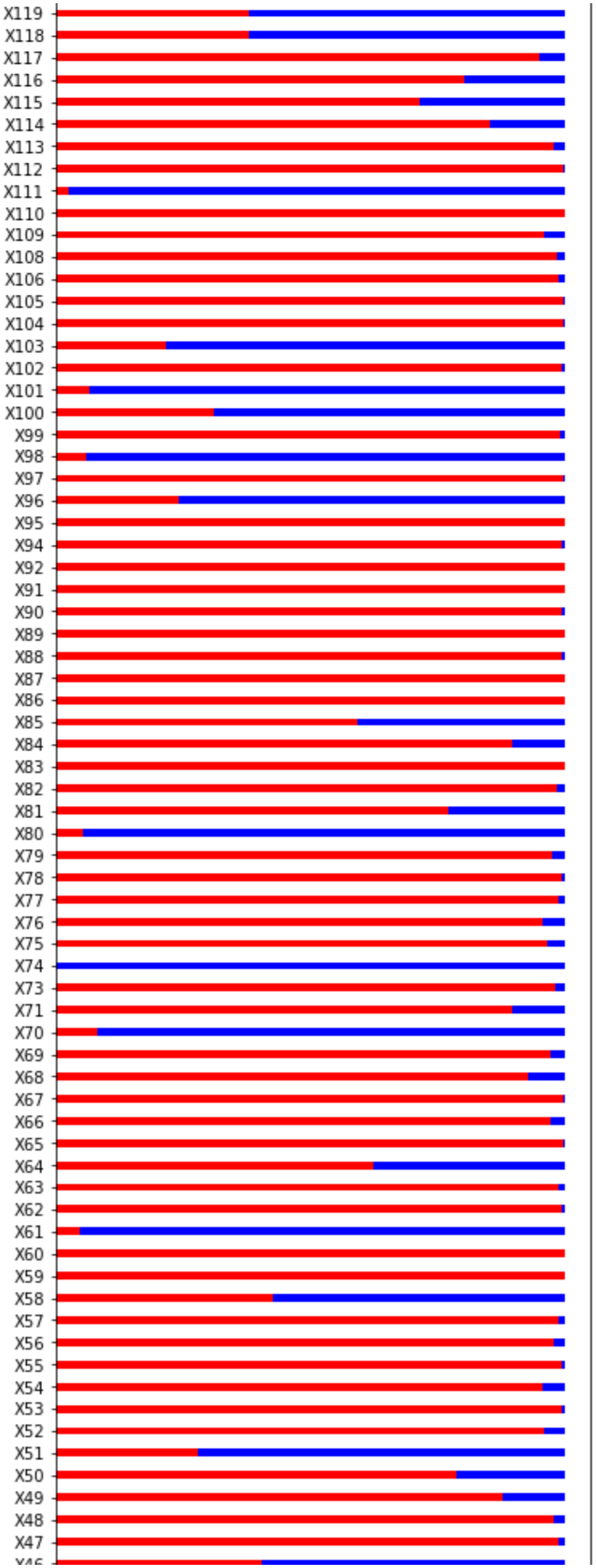
```
    one_list.append((df_train[col]==1).sum())
l = len(col_list)
arr = np.arange(l)
width = 0.35
plt.figure(figsize=(6,100))
plot_1 = plt.barh(arr, zero_list, width, color='red')
plot_2 = plt.barh(arr, one_list, width, left=zero_list, color="blue")
plt.yticks(arr, col_list)
plt.legend((plot_1[0], plot_2[0]), ('Zero count', 'One Count'))
plt.show()
```

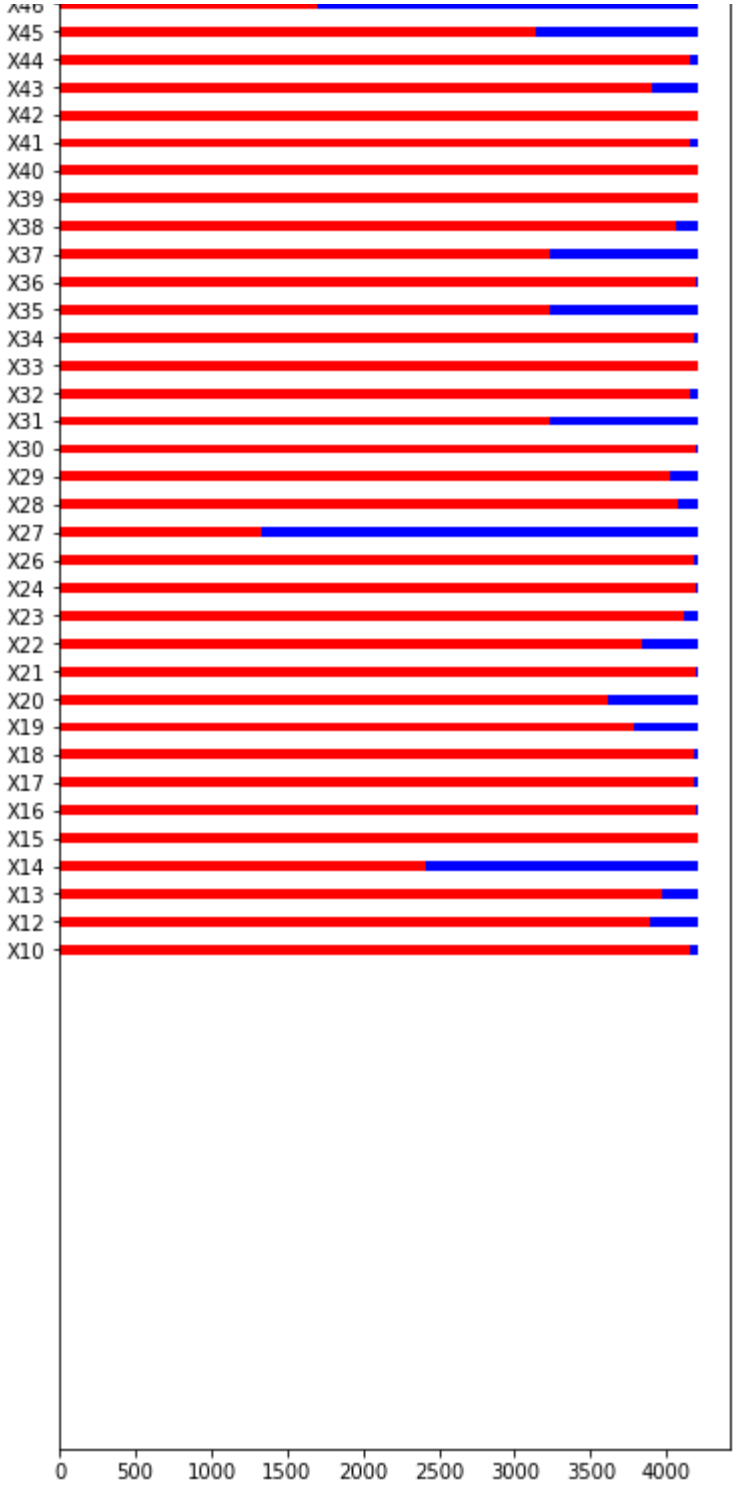



X343
X342
X341
X340
X339
X338
X337
X336
X335
X334
X333
X332
X331
X329
X328
X327
X326
X325
X324
X323
X322
X321
X320
X319
X318
X317
X316
X315
X314
X313
X312
X311
X310
X309
X308
X307
X306
X305
X304
X302
X301
X300
X299
X298
X296
X295
X294
X292
X291
X288
X287
X286
X285
X284
X283
X282
X281
X280
X279
X278
X277
X276
X275
X274
X273
X272
X271
X270
X269
X267


X266
X265
X264
X263
X262
X261
X260
X259
X258
X257
X256
X255
X254
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

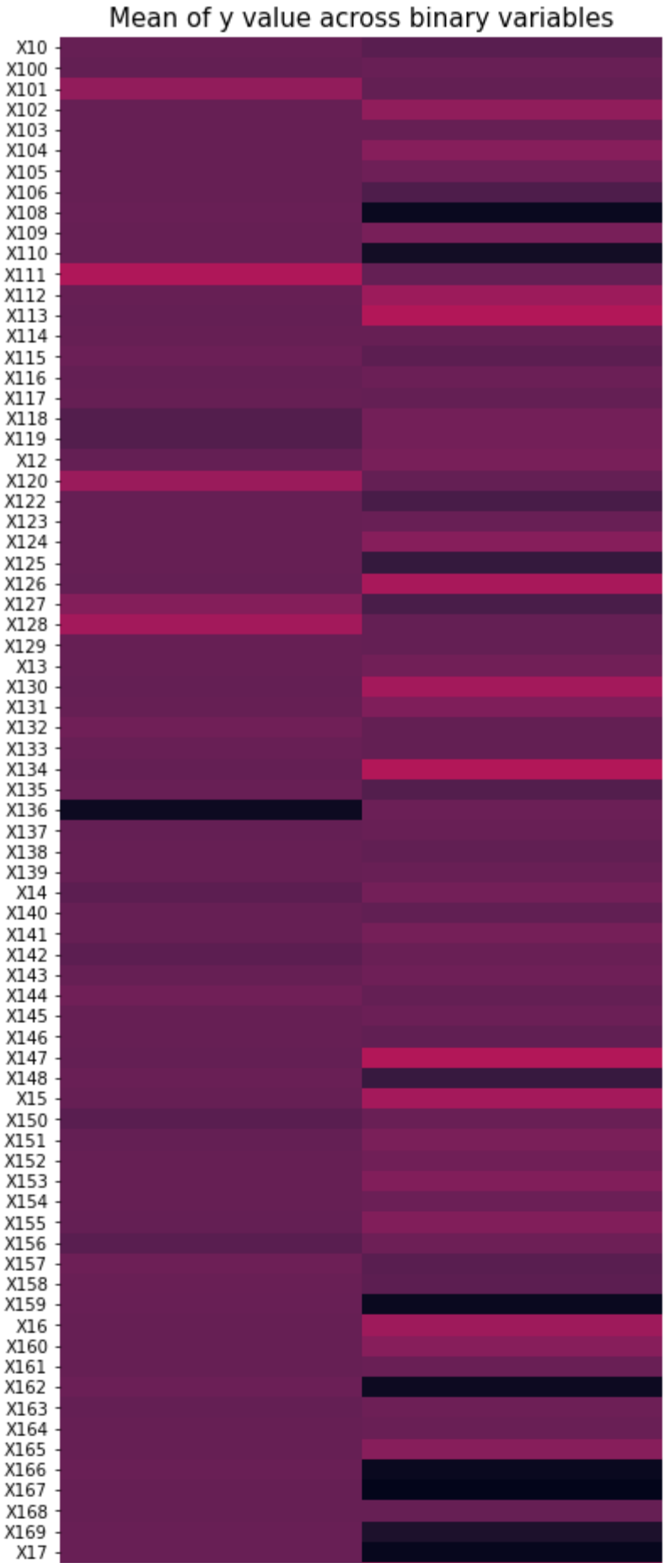

X119
X118
X117
X116
X115
X114
X113
X112
X111
X110
X109
X108
X106
X105
X104
X103
X102
X101
X100
X99
X98
X97
X96
X95
X94
X92
X91
X90
X89
X88
X87
X86
X85
X84
X83
X82
X81
X80
X79
X78
X77
X76
X75
X74
X73
X71
X70
X69
X68
X67
X66
X65
X64
X63
X62
X61
X60
X59
X58
X57
X56
X55
X54
X53
X52
X51
X50
X49
X48
X47

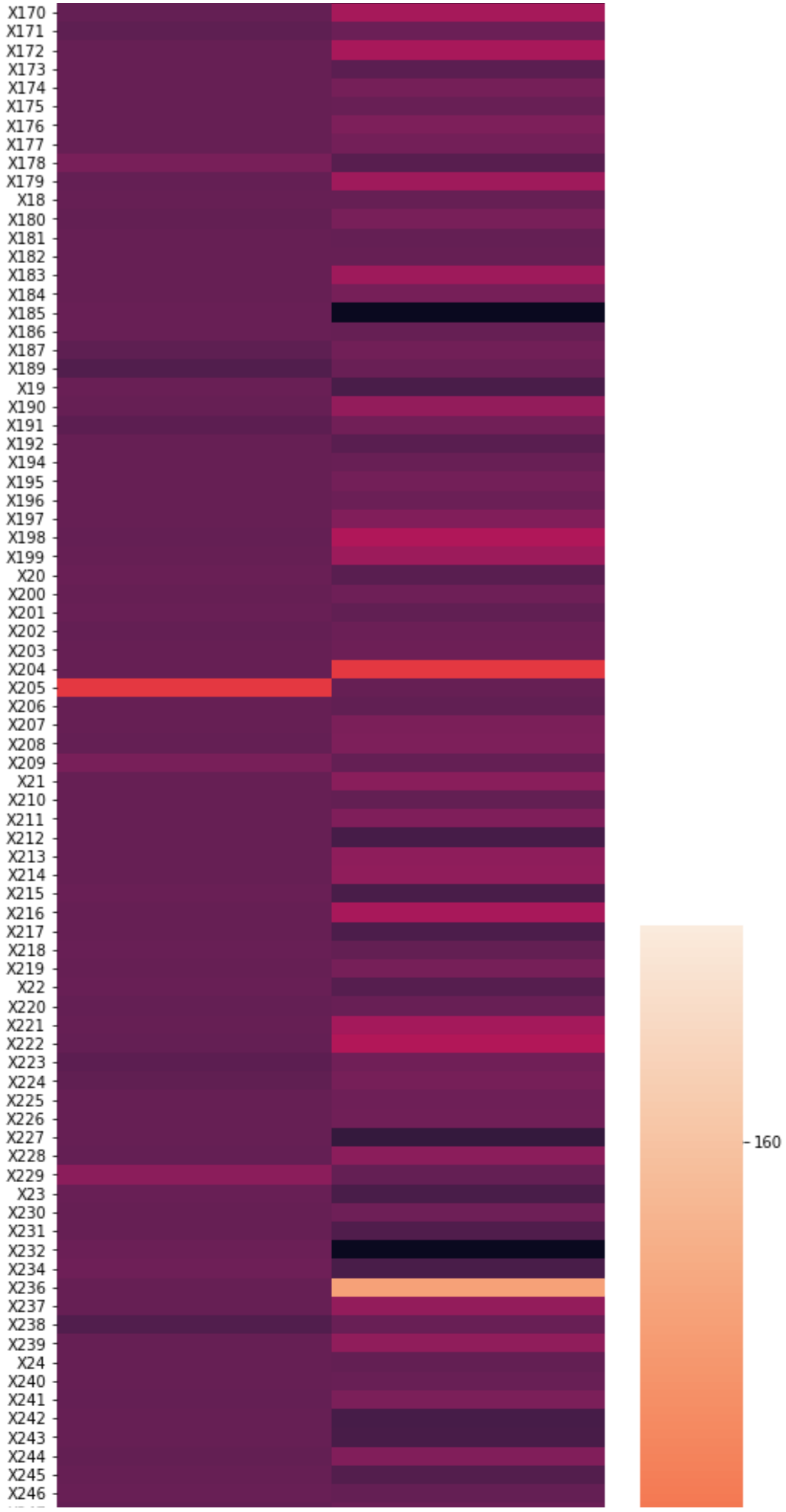


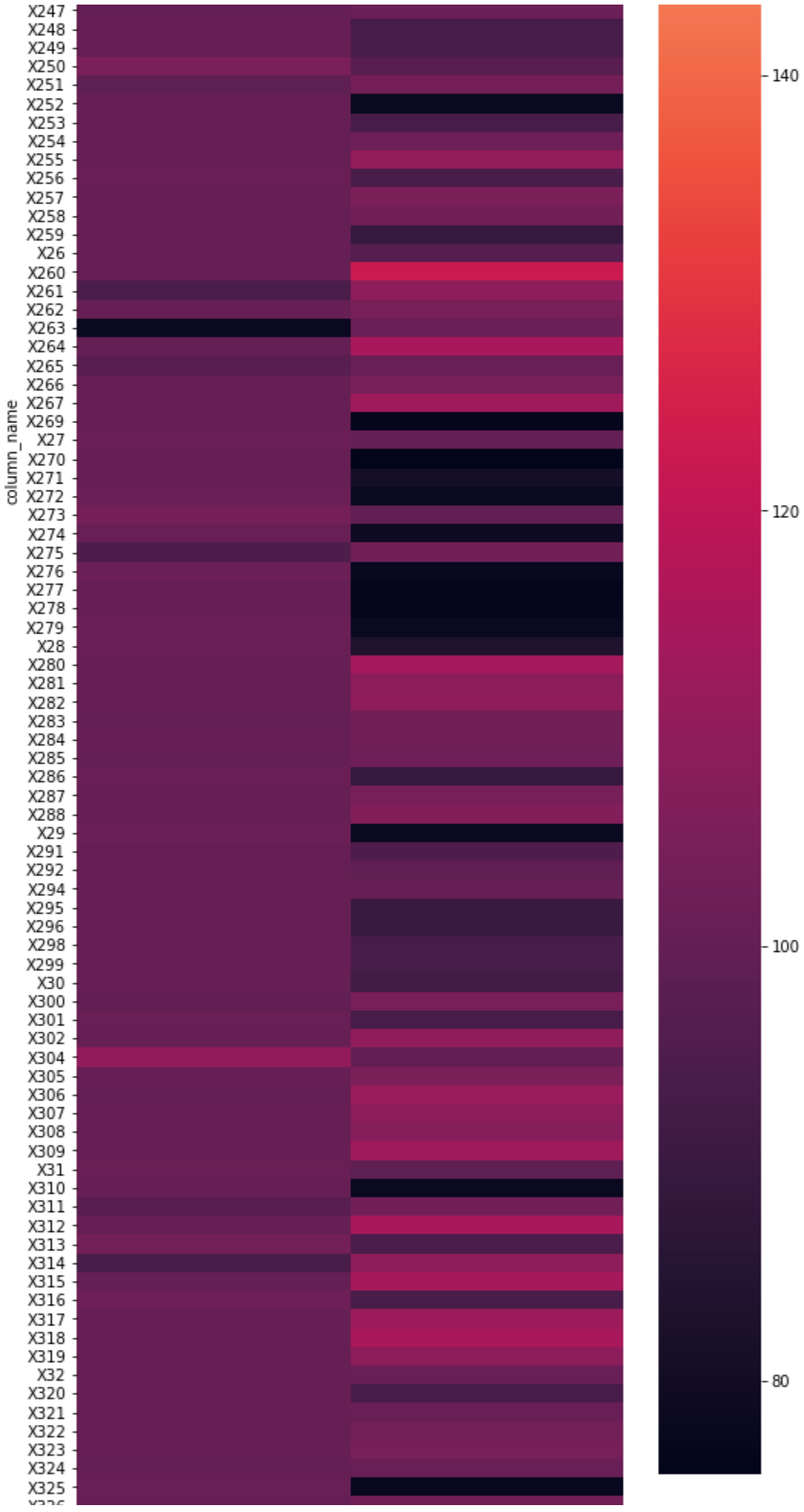
In [52]:

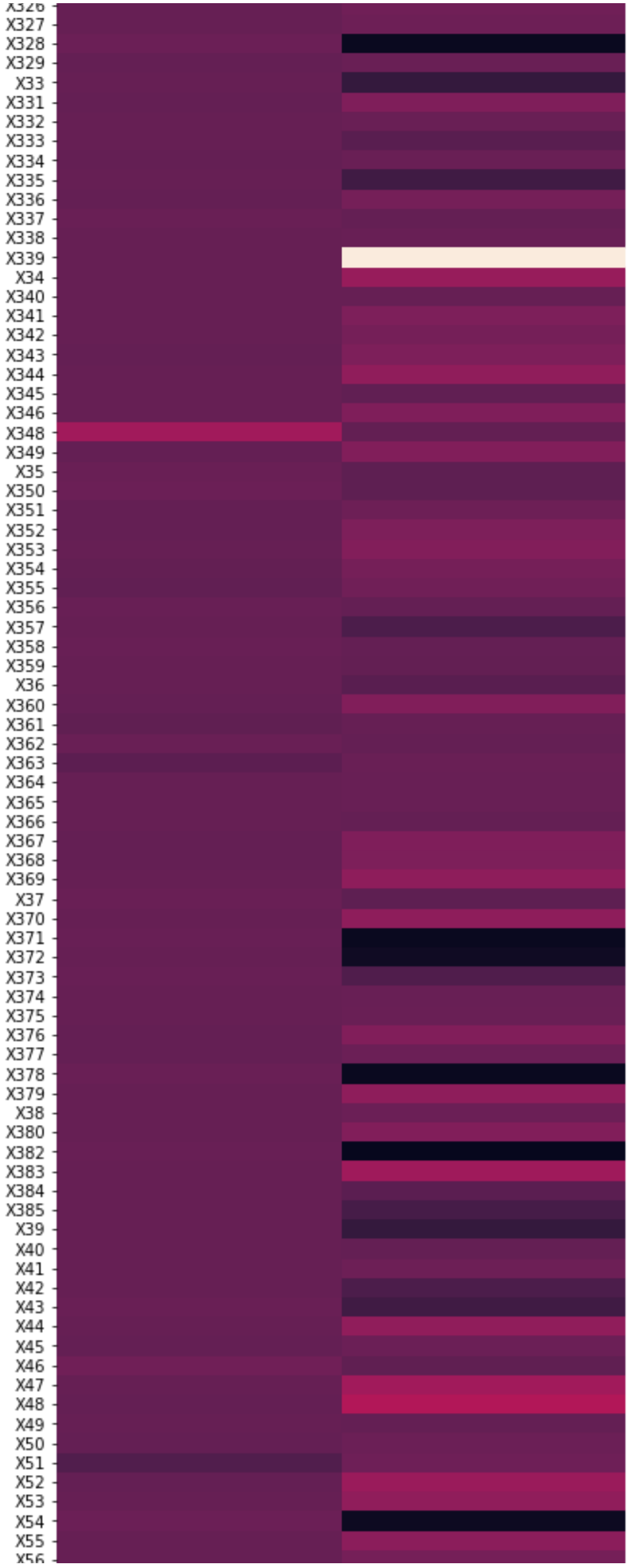
```
zero_mean_list = []
one_mean_list = []
cols_list = unique_value_dict['[0, 1]']
for col in cols_list:
    zero_mean_list.append(df_train.loc[df_train[col]==0].y.mean())
    one_mean_list.append(df_train.loc[df_train[col]==1].y.mean())
new_df = pd.DataFrame({"column_name":cols_list+cols_list, "value":[0]*len(cols_list) +
new_df = new_df.pivot('column_name', 'value', 'y_mean')

plt.figure(figsize=(8,80))
sns.heatmap(new_df)
```

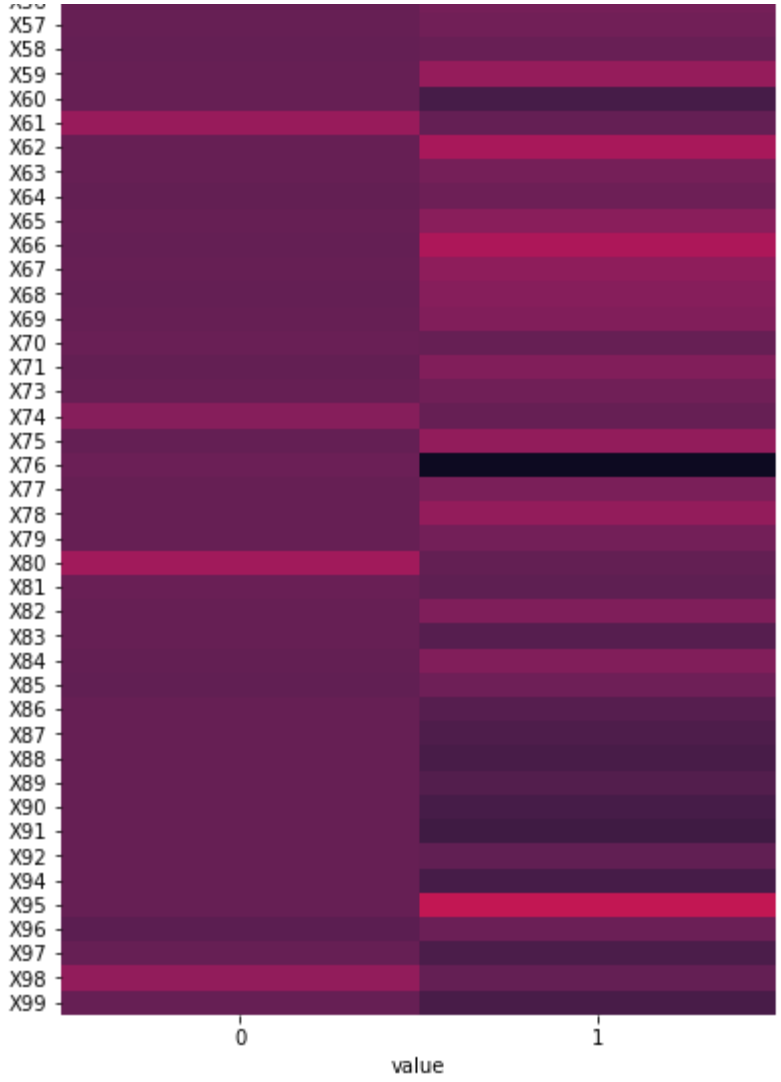
```
plt.title("Mean of y value across binary variables", fontsize=15)
plt.show()
```




X170
X171
X172
X173
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
160

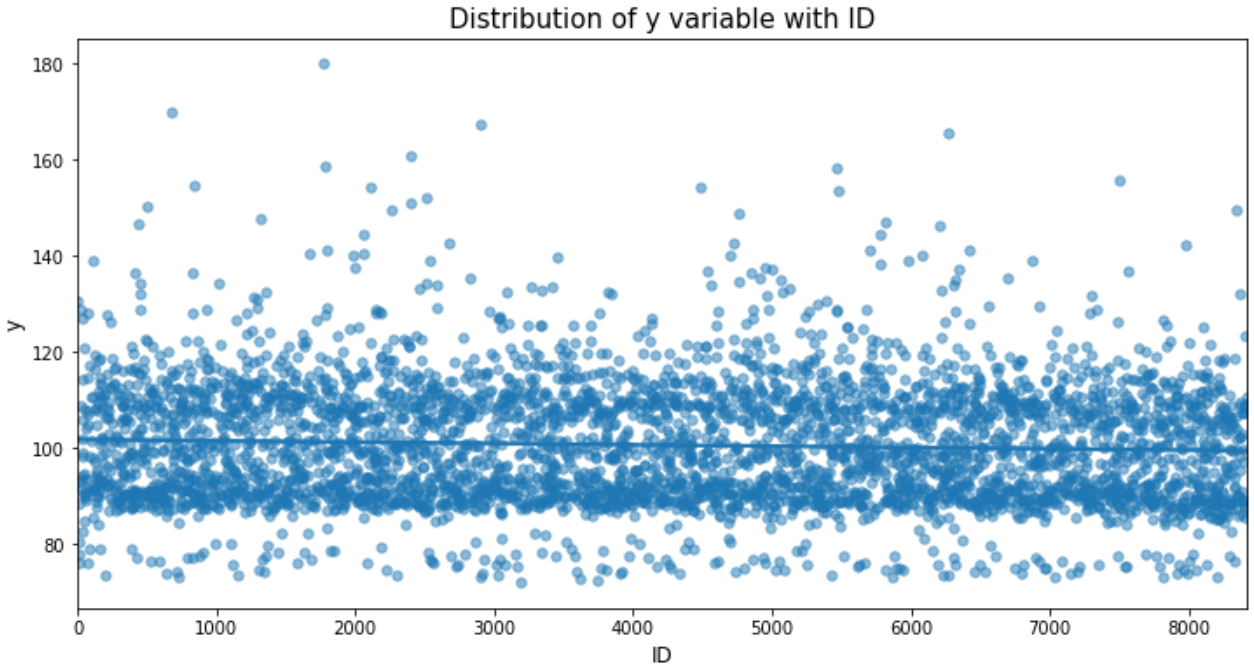

column_name
X247
X248
X249
X250
X251
X252
X253
X254
X255
X256
X257
X258
X259
X26
X260
X261
X262
X263
X264
X265
X266
X267
X269
X27
X270
X271
X272
X273
X274
X275
X276
X277
X278
X279
X28
X280
X281
X282
X283
X284
X285
X286
X287
X288
X29
X291
X292
X294
X295
X296
X298
X299
X30
X300
X301
X302
X304
X305
X306
X307
X308
X309
X31
X310
X311
X312
X313
X314
X315
X316
X317
X318
X319
X32
X320
X321
X322
X323
X324
X325
140
120
100
80


X326
X327
X328
X329
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



In [53]:

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

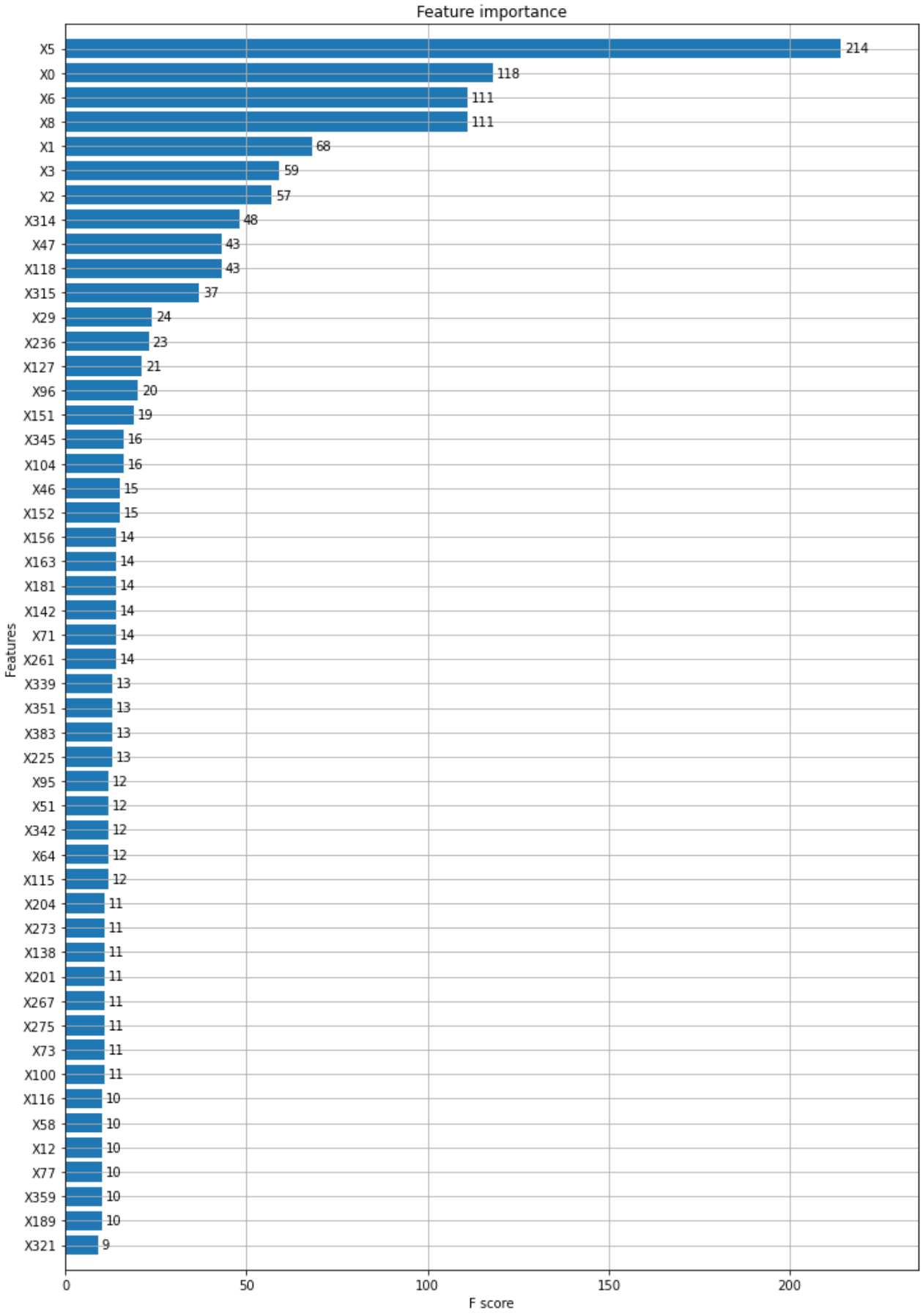


In [54]:

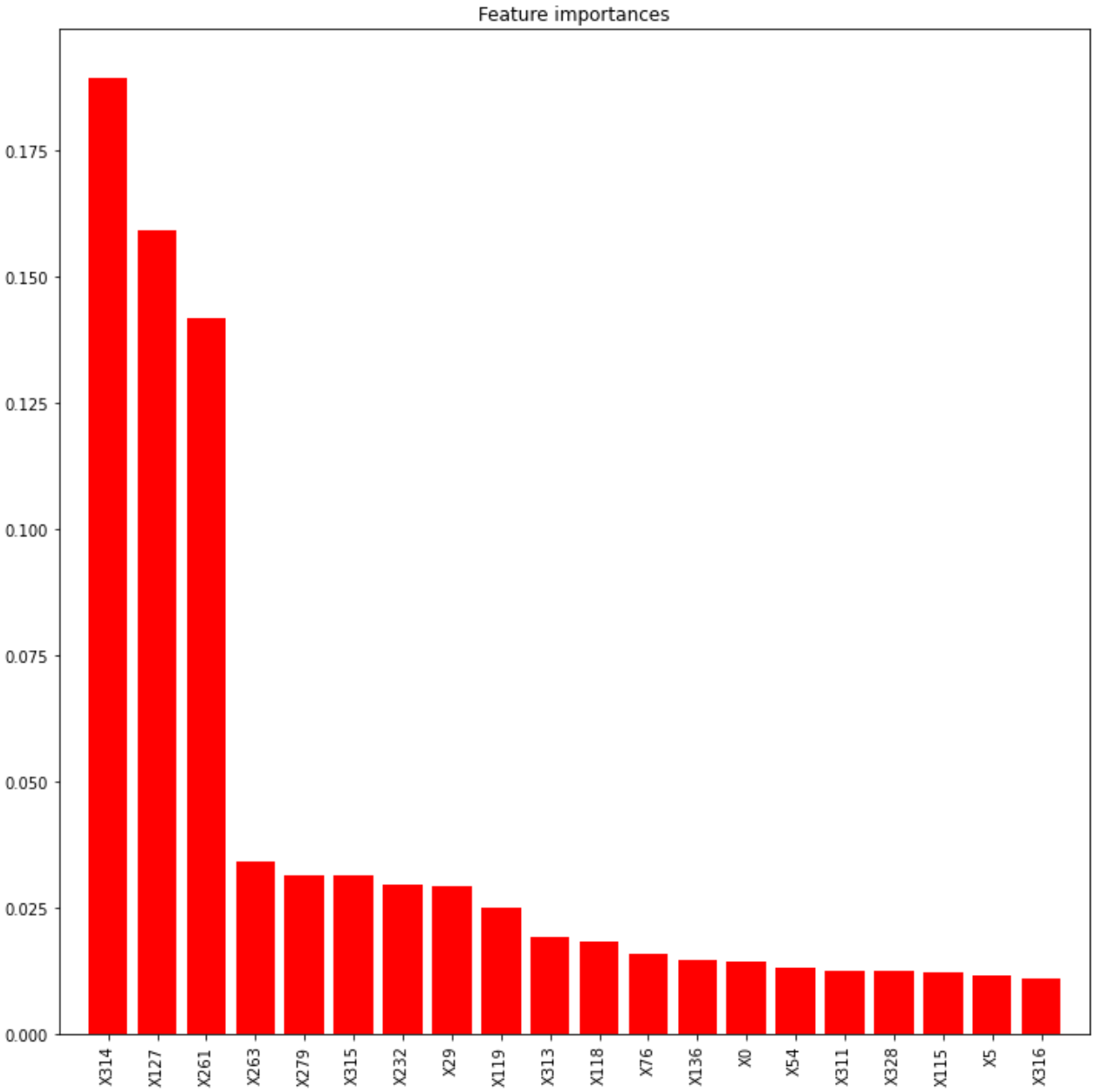
```
for f in ["X0", "X1", "X2", "X3", "X4", "X5", "X6", "X8"]:
        lbl = preprocessing.LabelEncoder()
        lbl.fit(list(df_train[f].values))
        df_train[f] = lbl.transform(list(df_train[f].values))

train_y = df_train['y'].values
train_X =df_train.drop(["ID", "y"], 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

# 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()
```

```
[23:50:37] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/src/obj
ective/regression_obj.cu:171: reg:linear is now deprecated in favor of reg:squarederror.
[23:50:37] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/src/lea
rner.cc:573:
Parameters: { "silent" } might not be used.

  This may not be accurate due to some parameters are only used in language bindings but
  passed down to XGBoost core.  Or some parameters are not used but slip through this
```

```
verification. Please open an issue if you find above cases.
```



In [55]:
```
from sklearn import ensemble
model = ensemble.RandomForestRegressor(n_estimators=200, max_depth=10, min_samples_leaf
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()
```



In [ ]:
```
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: