```
from google.colab import drive
drive.mount('/content/drive')
```

Run this cell to mount your Google Drive.

DISMISS

```
browser: https://accounts.google.com/o/oauth2/auth?client_
ation code:
··········
Mounted at /content/drive
```

```
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
#print(check_output(["ls", "../input"]).decode("utf8"))
```

```
train_df = pd.read_csv("/content/drive/My Drive/Projects for Submission/Project 1 - Merce
test_df = pd.read_csv("/content/drive/My Drive/Projects for Submission/Project 1 - Mercec
print("Train shape : ", train_df.shape)
print("Test shape : ", test_df.shape)
```

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

```
train_df.head()
```

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

```
test_df.head()
```

	ID	X0	X1	X2	X3	X4	X5	X6	X8	X10	X11	X12	X13	X14	X15	X16	X17	X18
0	1	az	v	n	f	d	t	a	w	0	0	0	0	0	0	0	0	0
1	2	t	b	ai	a	d	b	g	y	0	0	0	0	0	0	0	0	0
					f	d	a	j	j	0	0	0	0	1	0	0	0	0
					f	d	z	l	n	0	0	0	0	0	0	0	0	0
4	5	w	s	as	c	d	y	i	m	0	0	0	0	1	0	0	0	0

Run this cell to mount your Google Drive.

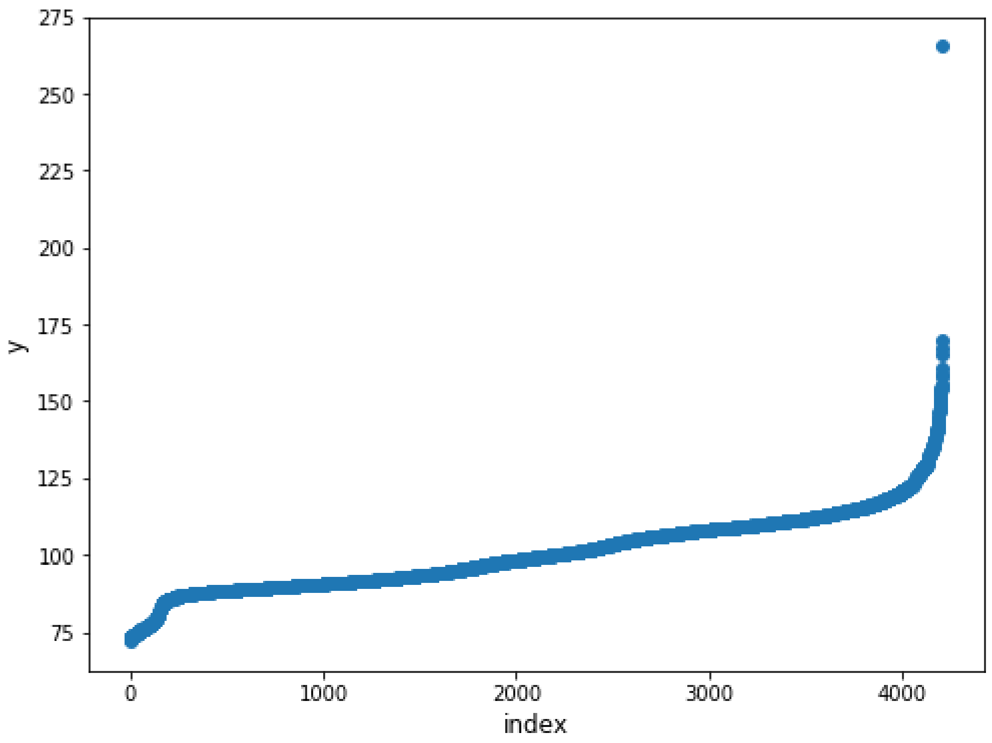
DISMISS

```
y = train_df
```

```
train_df.columns
```

```
Index(['Id', 'v2a1', 'hacdor', 'rooms', 'hacapo', 'v14a', 'refrig', 'v18q',
        'v18q1', 'r4h1',
        ...
        'SQBescolari', 'SQBage', 'SQBhogar_total', 'SQBedjefe', 'SQBhogar_nin',
        'SQBovercrowding', 'SQBdependency', 'SQBmeaned', 'agesq', 'Target'],
       dtype='object', length=143)
```

```
plt.figure(figsize=(8,6))
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()
```



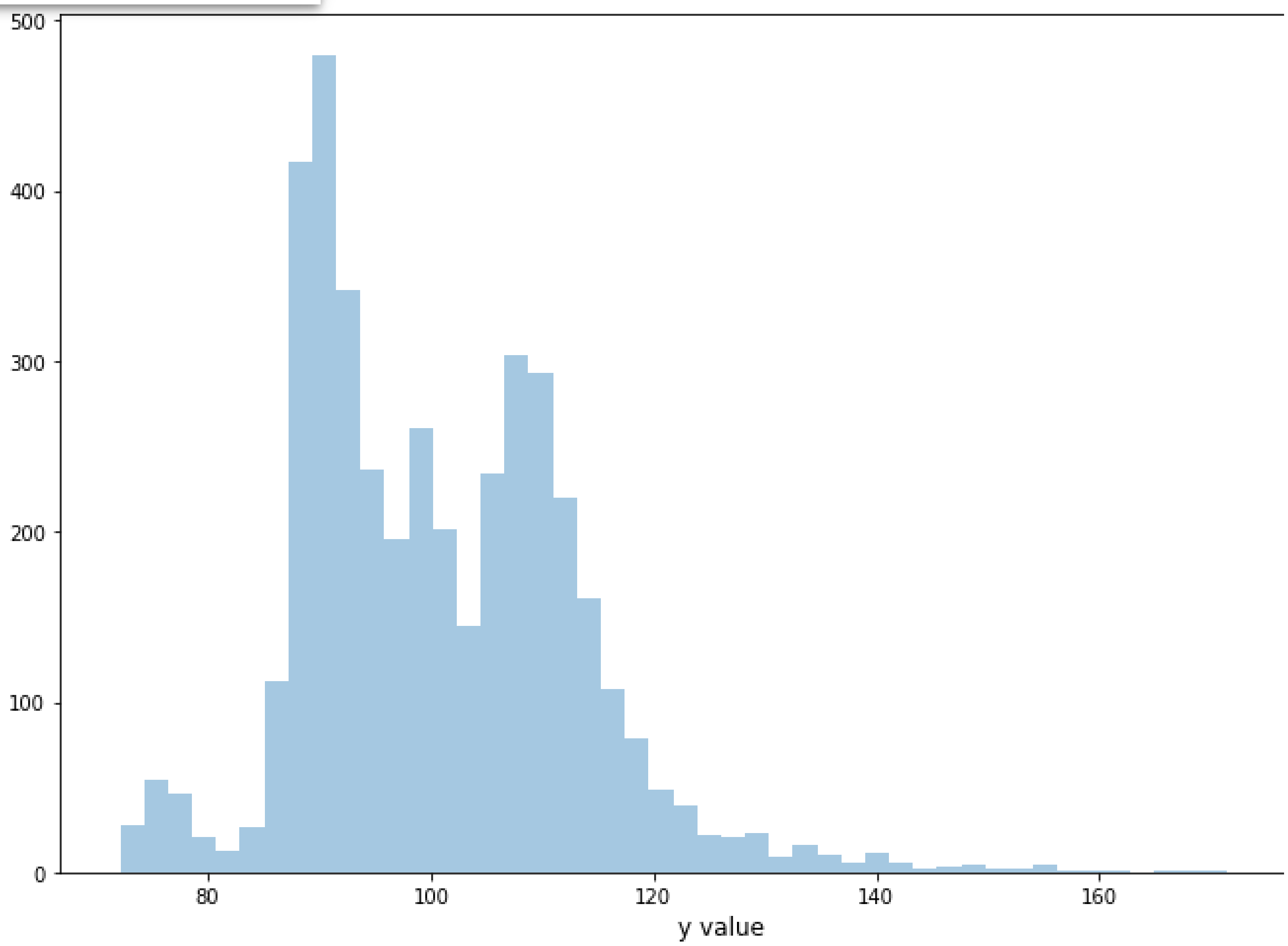
```
ulimit = 180
train_df['y'].ix[train_df['y']>ulimit] = ulimit

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

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: DeprecationWar
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing
```

Run this cell to mount your Google Drive.

DISMISS

```
                                   on here:
                                   a.org/pandas-docs/stable/indexing.html#ix-indexer-is-deprec
```



```
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.ix[:10,:]
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:1: DeprecationWar
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
```

Run this cell to mount your Google Drive.

DISMISS

```
.org/pandas-docs/stable/indexing.html#ix-indexer-is-deprec
 launching an IPython kernel.
```

		ype
0	ID	int64
1	y	float64
2	X0	object
3	X1	object
4	X2	object
5	X3	object
6	X4	object
7	X5	object
8	X6	object
9	X8	object
10	X10	int64

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

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:3: DeprecationWar
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/indexing.html#ix-indexer-is-deprec
  This is separate from the ipykernel package so we can avoid doing imports un
```

column_name	missing_count

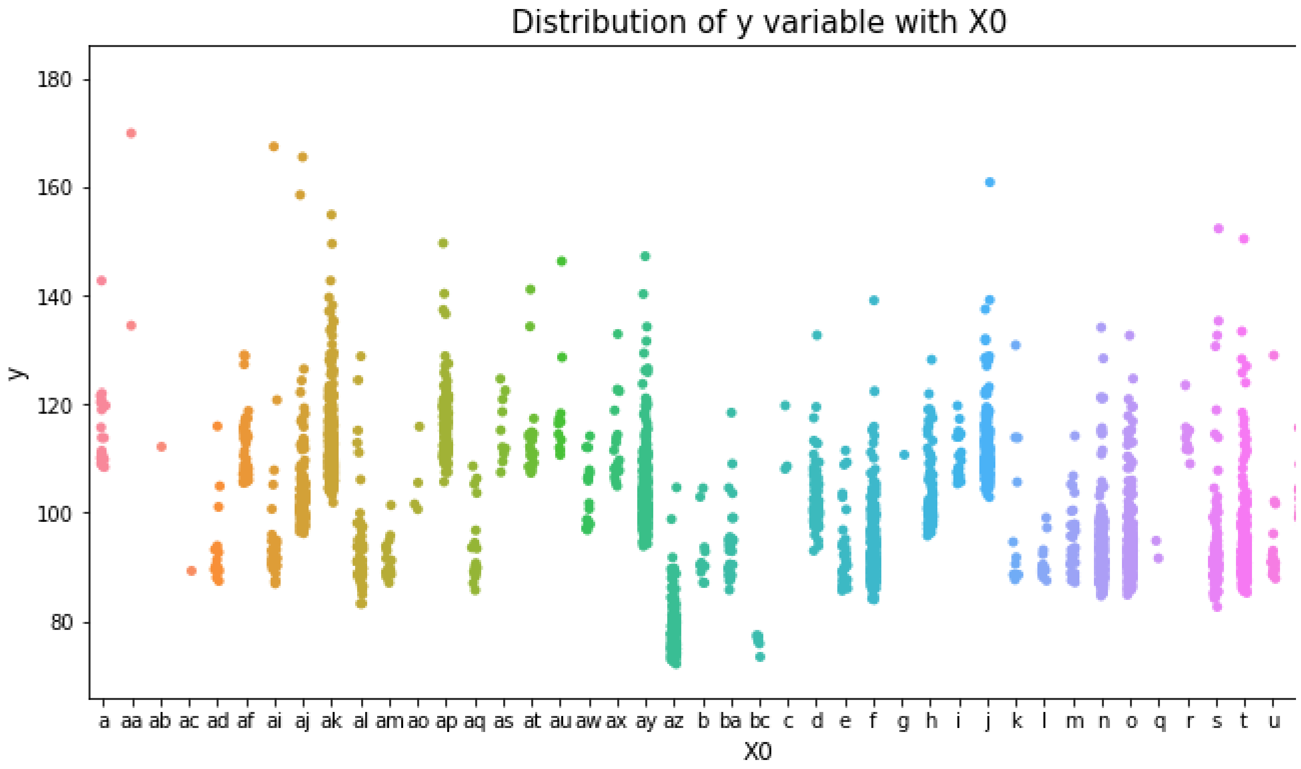
```
unique_values_dict = {}
for col in train_df.columns:
    if col not in ["ID", "y", "X0", "X1", "X2", "X3", "X4", "X5", "X6", "X8"]:
        unique_value = str(np.sort(train_df[col].unique()).tolist())
        tlist = unique_values_dict.get(unique_value, [])
        tlist.append(col)
        unique_values_dict[unique_value] = 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',
--------------------------------------------------
Columns containing the unique values :  [0]
                                 '', 'X233', 'X235', 'X268', 'X289', 'X290', 'X293', 'X297',
                                 --------------------------------
```

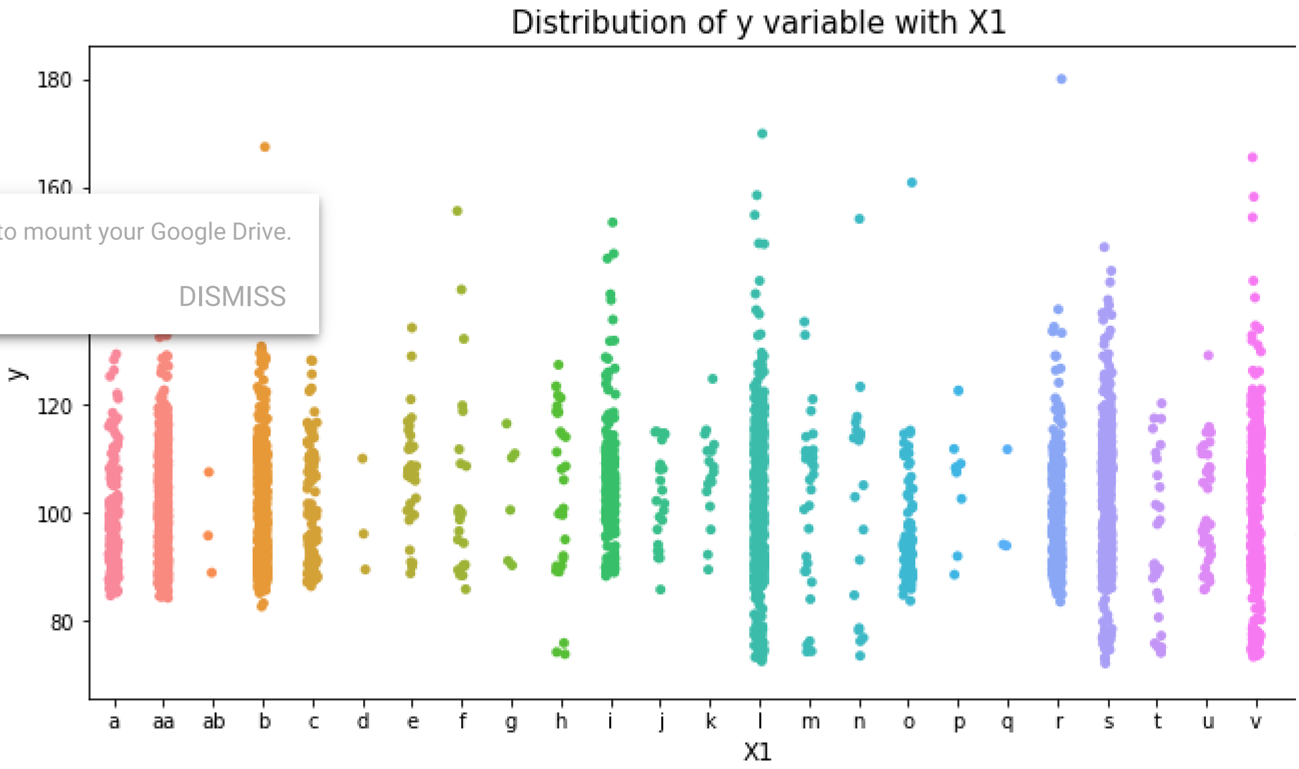
Run this cell to mount your Google Drive.

DISMISS

```
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()
```



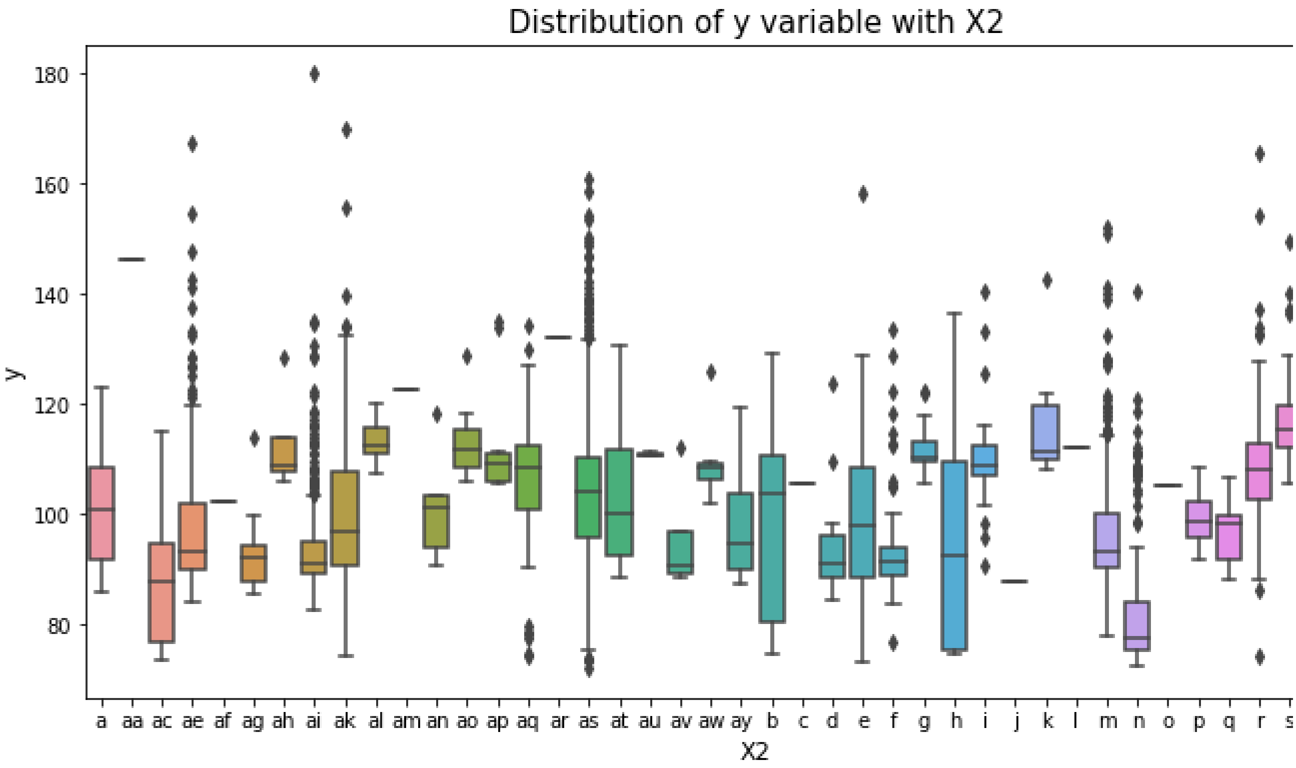
```
var_name = "X1"
col_order = np.sort(train_df[var_name].unique()).tolist()
plt.figure(figsize=(12,6))
sns.stripplot(x=var_name, y='y', data=train_df, order=col_order)
plt.xlabel(var_name, fontsize=12)
plt.ylabel('y', fontsize=12)
plt.title("Distribution of y variable with "+var_name, fontsize=15)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



Run this cell to mount your Google Drive.

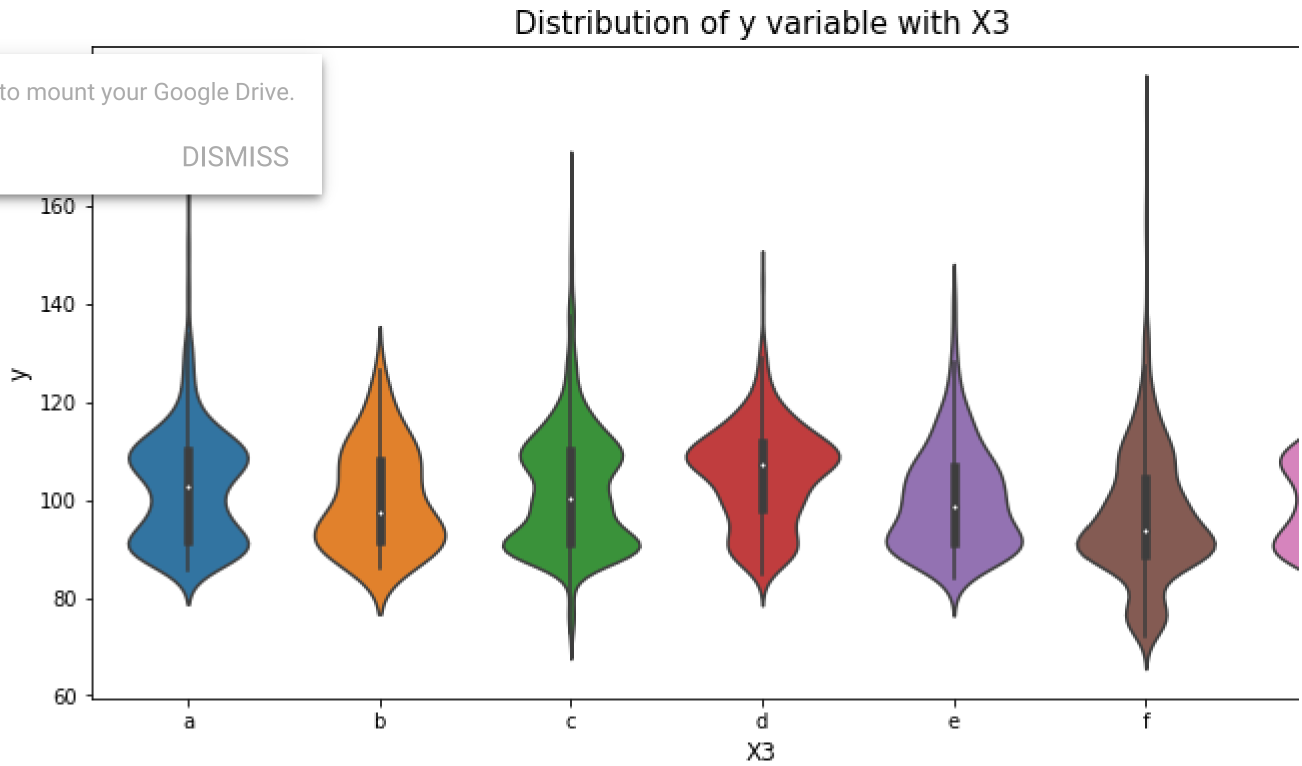
DISMISS

```
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()
```



```
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)
```

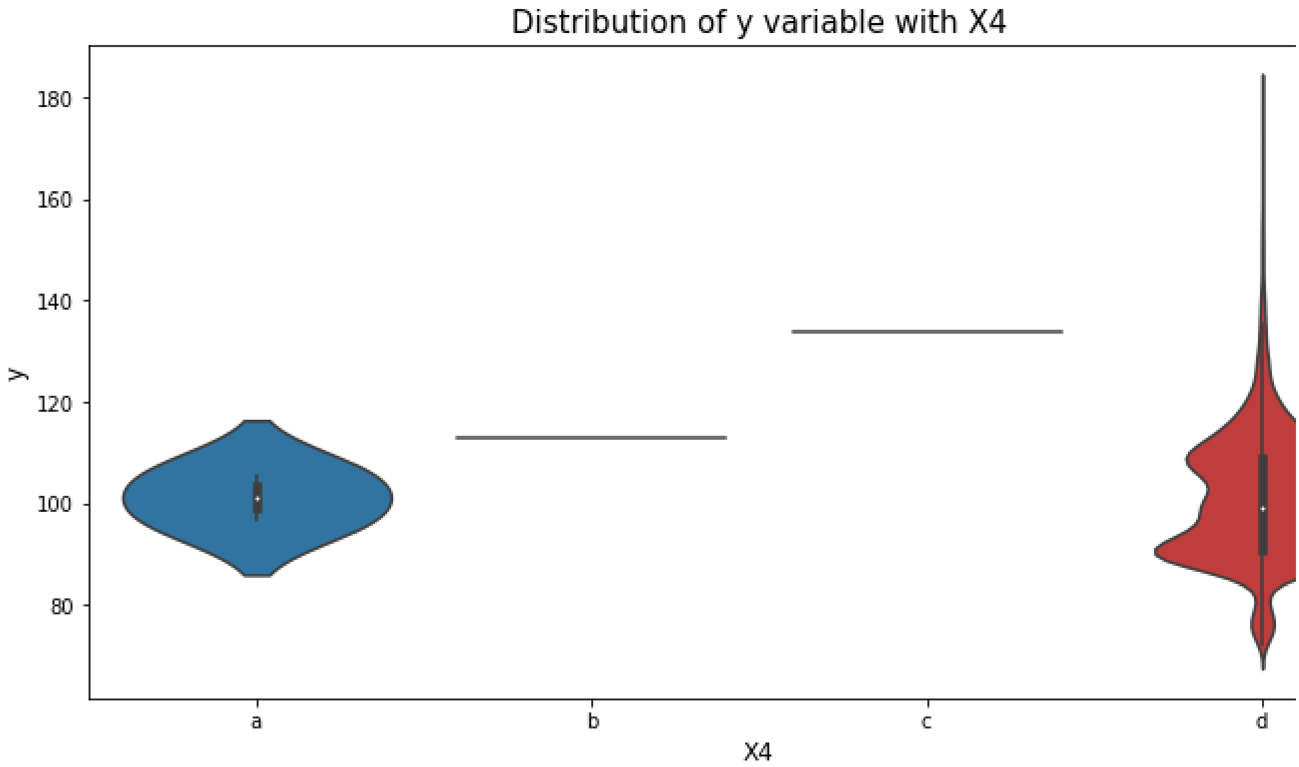
```
    plt.xlabel(var_name, fontsize=12)
    plt.ylabel('y', fontsize=12)
    plt.title("Distribution of y variable with "+var_name, fontsize=15)
    plt.show()
```


Distribution of y variable with X3

Run this cell to mount your Google Drive.

DISMISS

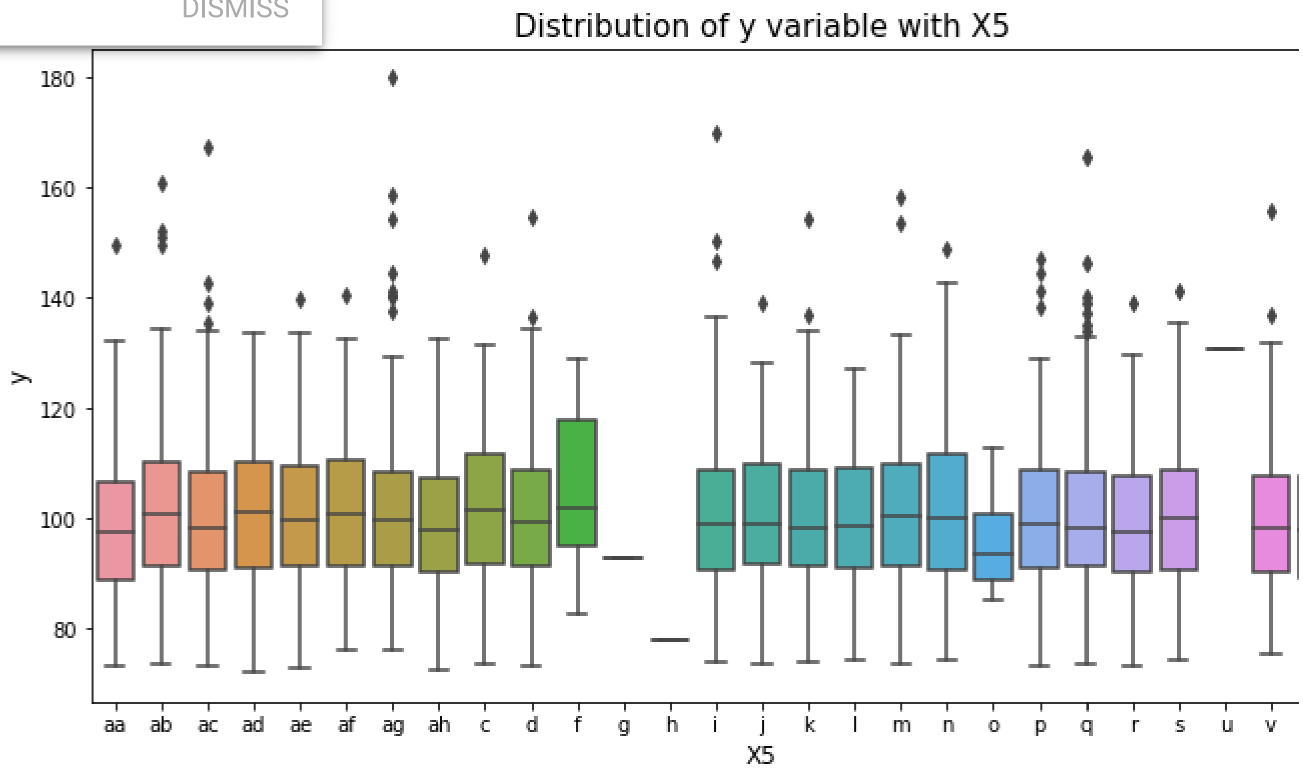
```
    var_name = "X4"
    col_order = np.sort(train_df[var_name].unique()).tolist()
    plt.figure(figsize=(12,6))
    sns.violinplot(x=var_name, y='y', data=train_df, order=col_order)
    plt.xlabel(var_name, fontsize=12)
    plt.ylabel('y', fontsize=12)
    plt.title("Distribution of y variable with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


Distribution of y variable with X4

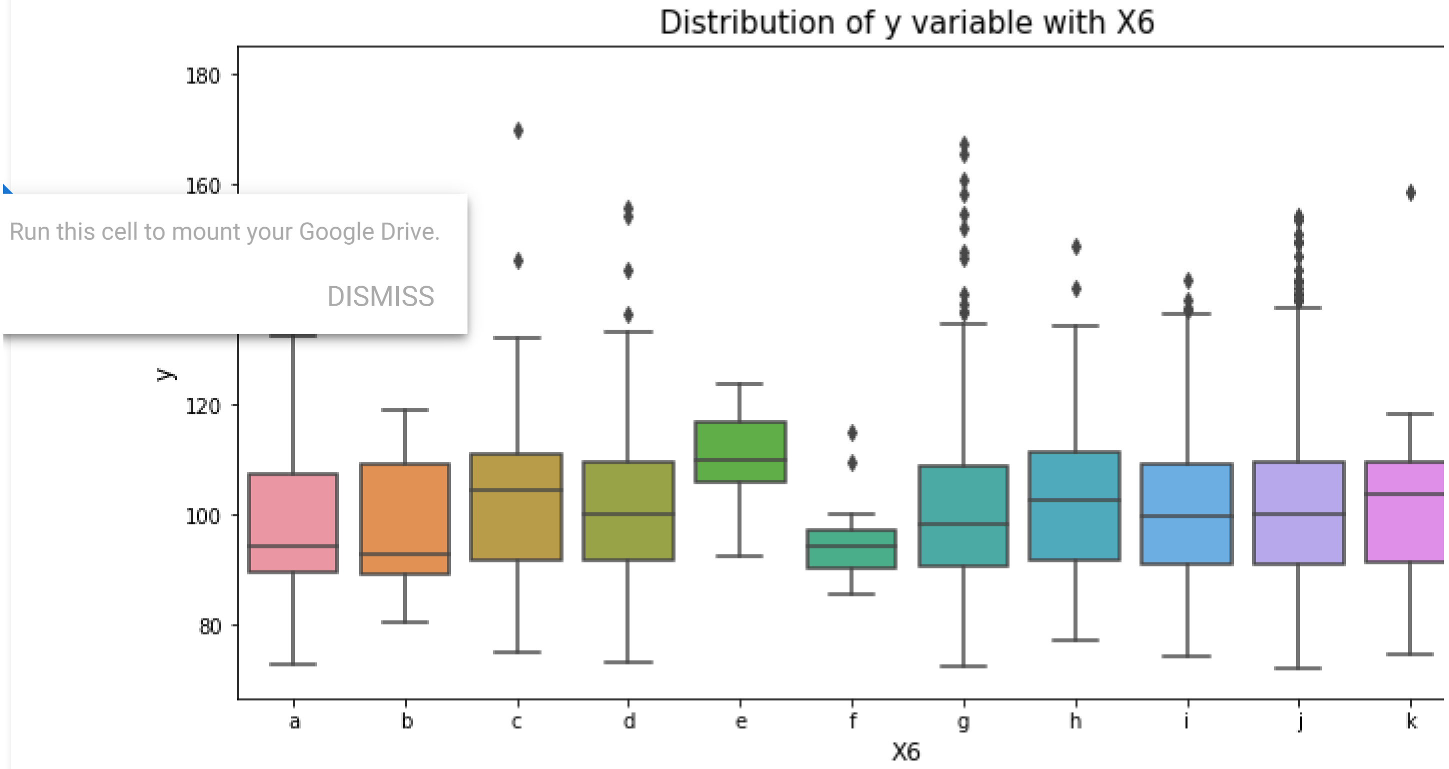
```
var_name = "X5"
col_order = np.sort(train_df[var_name].unique()).tolist()
plt.figure(figsize=(12,6))
sns.boxplot(x=var_name, y='y', data=train_df, order=col_order)
plt.xlabel(var_name, fontsize=12)
plt.ylabel('y', fontsize=12)
plt.title("Distribution of y variable with "+var_name, fontsize=15)
```

Run this cell to mount your Google Drive.

DISMISS



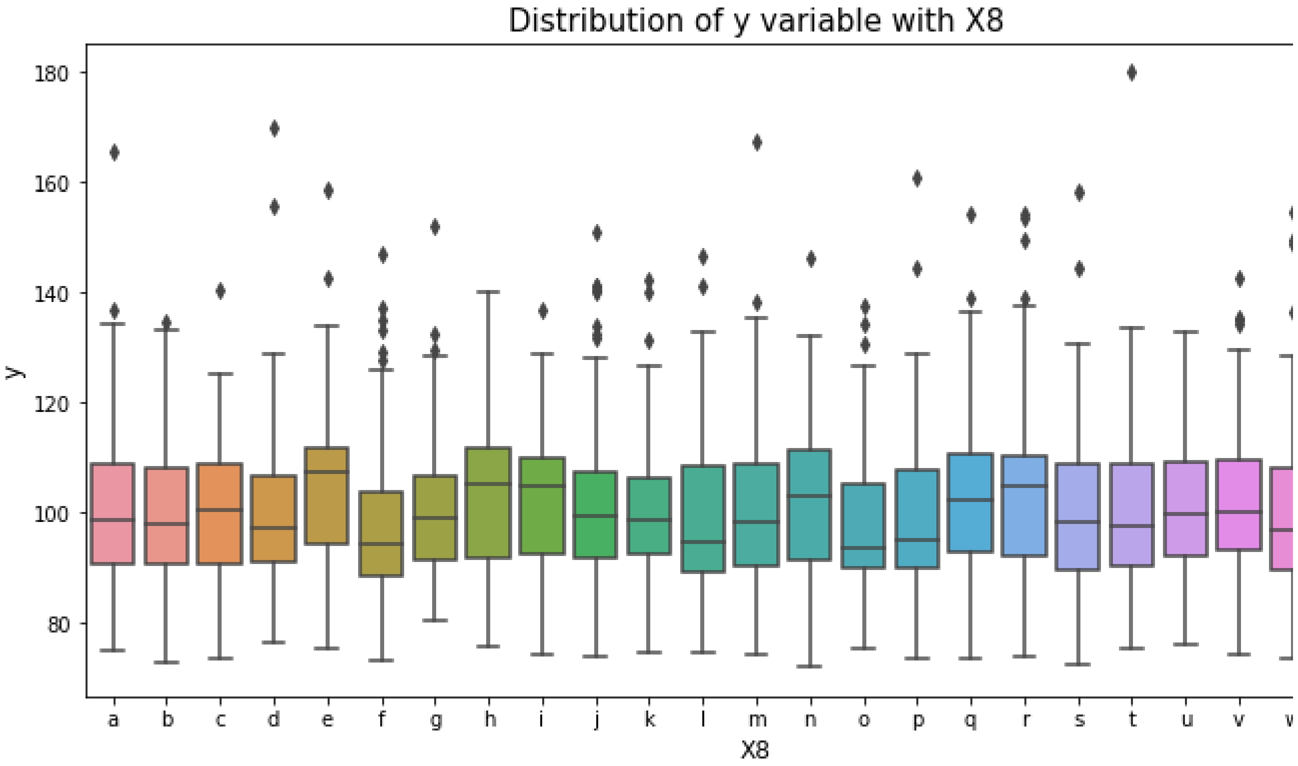
```
var_name = "X6"
col_order = np.sort(train_df[var_name].unique()).tolist()
plt.figure(figsize=(12,6))
sns.boxplot(x=var_name, y='y', data=train_df, order=col_order)
plt.xlabel(var_name, fontsize=12)
plt.ylabel('y', fontsize=12)
plt.title("Distribution of y variable with "+var_name, fontsize=15)
plt.show()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



Run this cell to mount your Google Drive.

DISMISS

```
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()
```



```
zero_count_list = []
one_count_list = []
cols_list = unique_values_dict['[0, 1]']
```

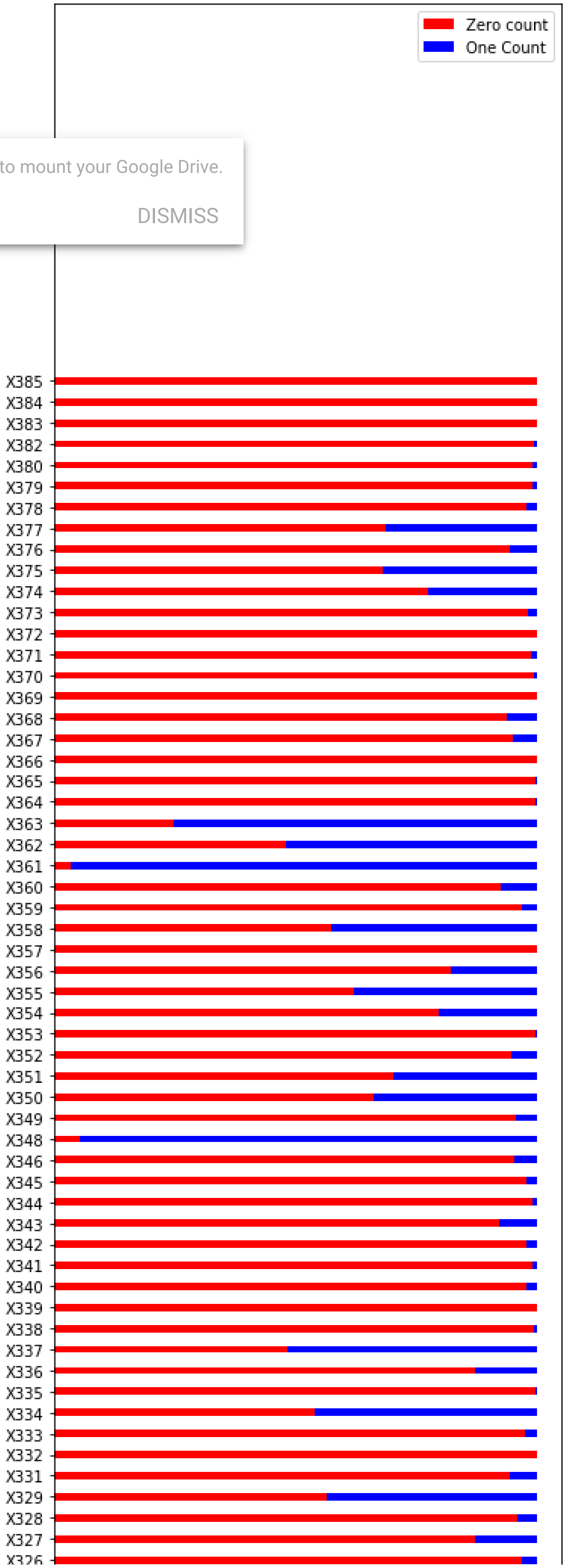
```
for col in cols_list:
    zero_count_list.append((train_df[col]==0).sum())
    one_count_list.append((train_df[col]==1).sum())

N = len(cols_list)
ind = np.arange(N)
width = 0.35

                                    )
                                    unt_list, width, color='red')
                                    nt_list, width, left=zero_count_list, color="blue")

plt.legend((p1[0], p2[0]), ('Zero count', 'One Count'))
plt.show()
```

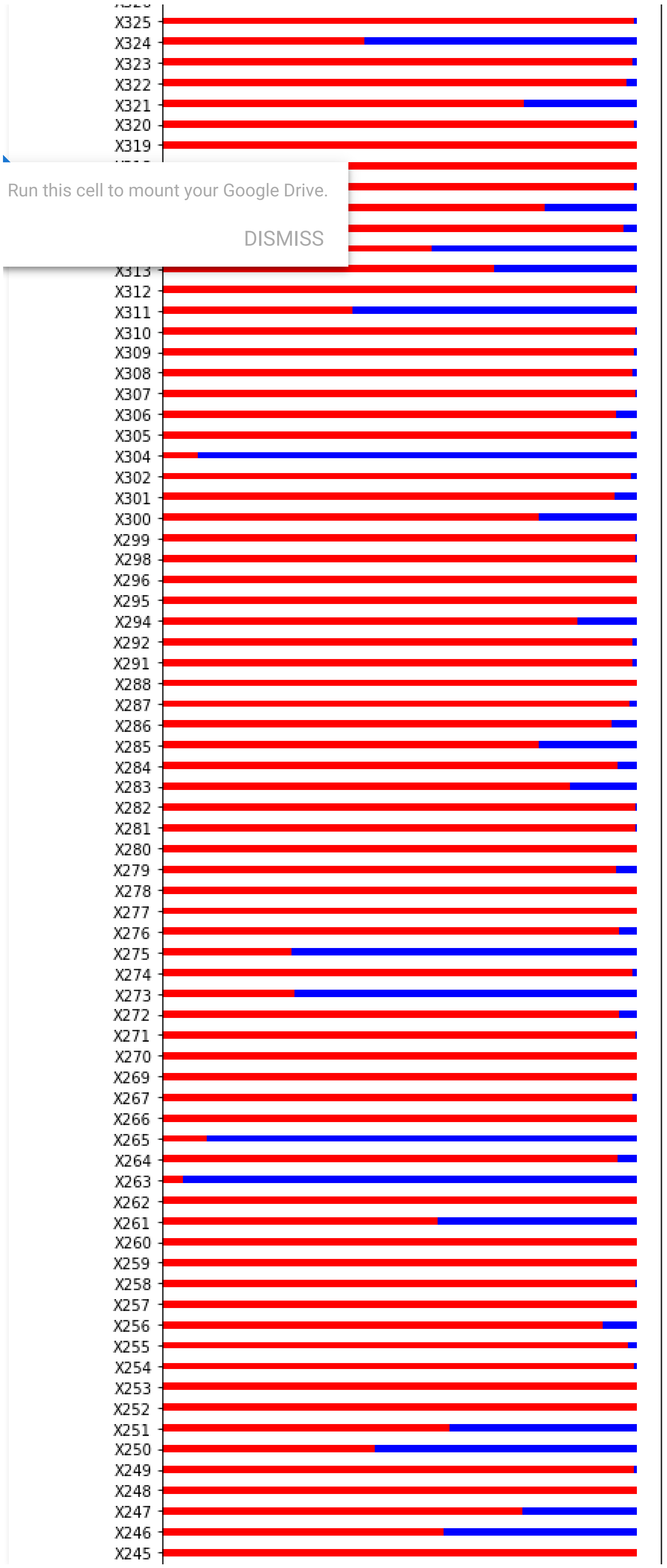
Run this cell to mount your Google Drive.

DISMISS

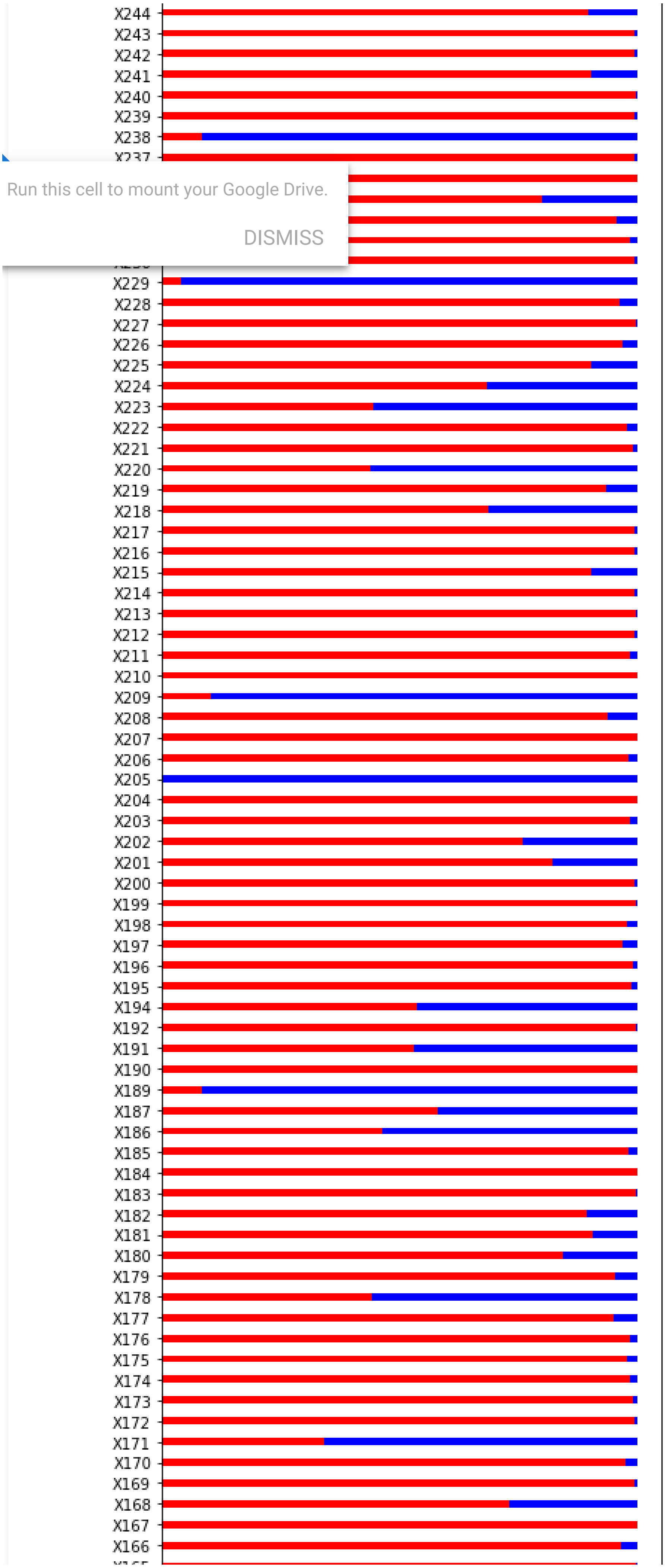
Run this cell to mount your Google Drive.

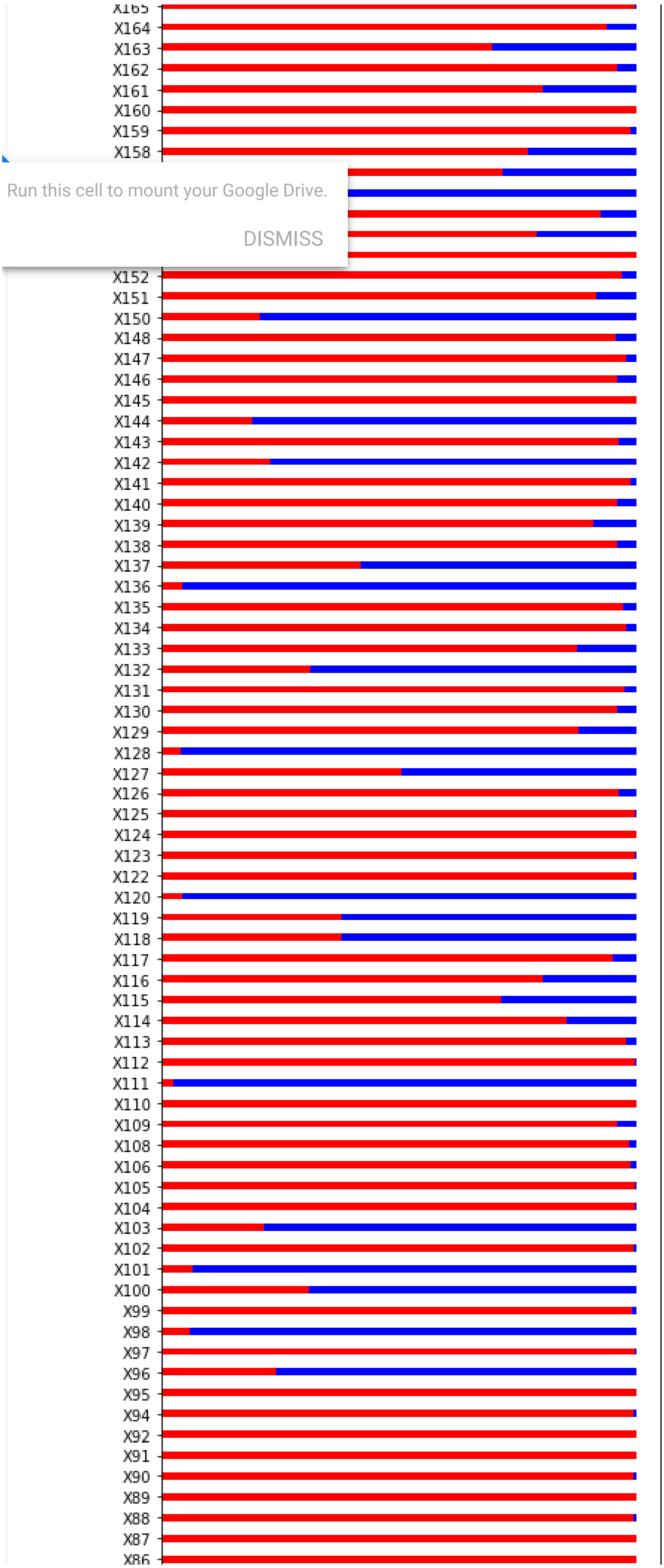
DISMISS





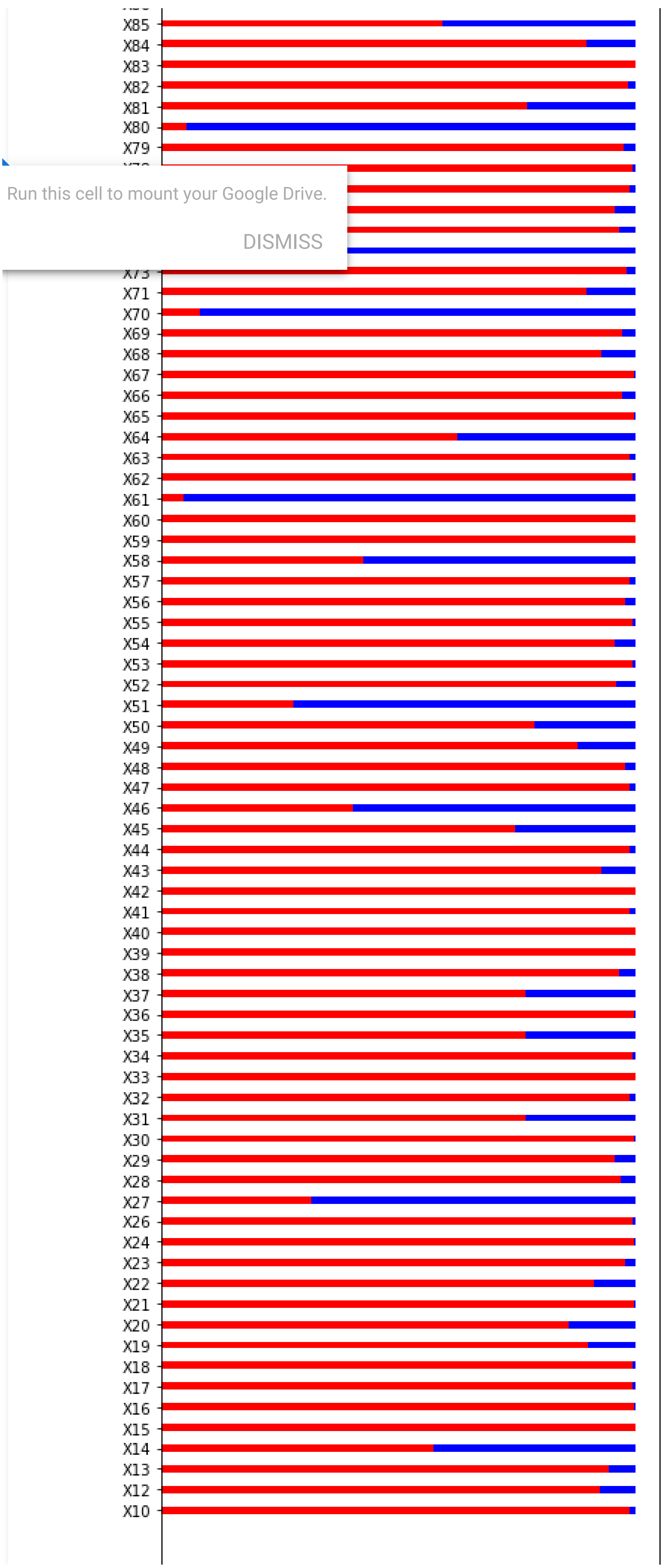
Run this cell to mount your Google Drive.


X244
X243
X242
X241
X240
X239
X238
X237
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
Run this cell to mount your Google Drive.
DISMISS



Run this cell to mount your Google Drive.

DISMISS



Run this cell to mount your Google Drive.

DISMISS

Run this cell to mount your Google Drive.

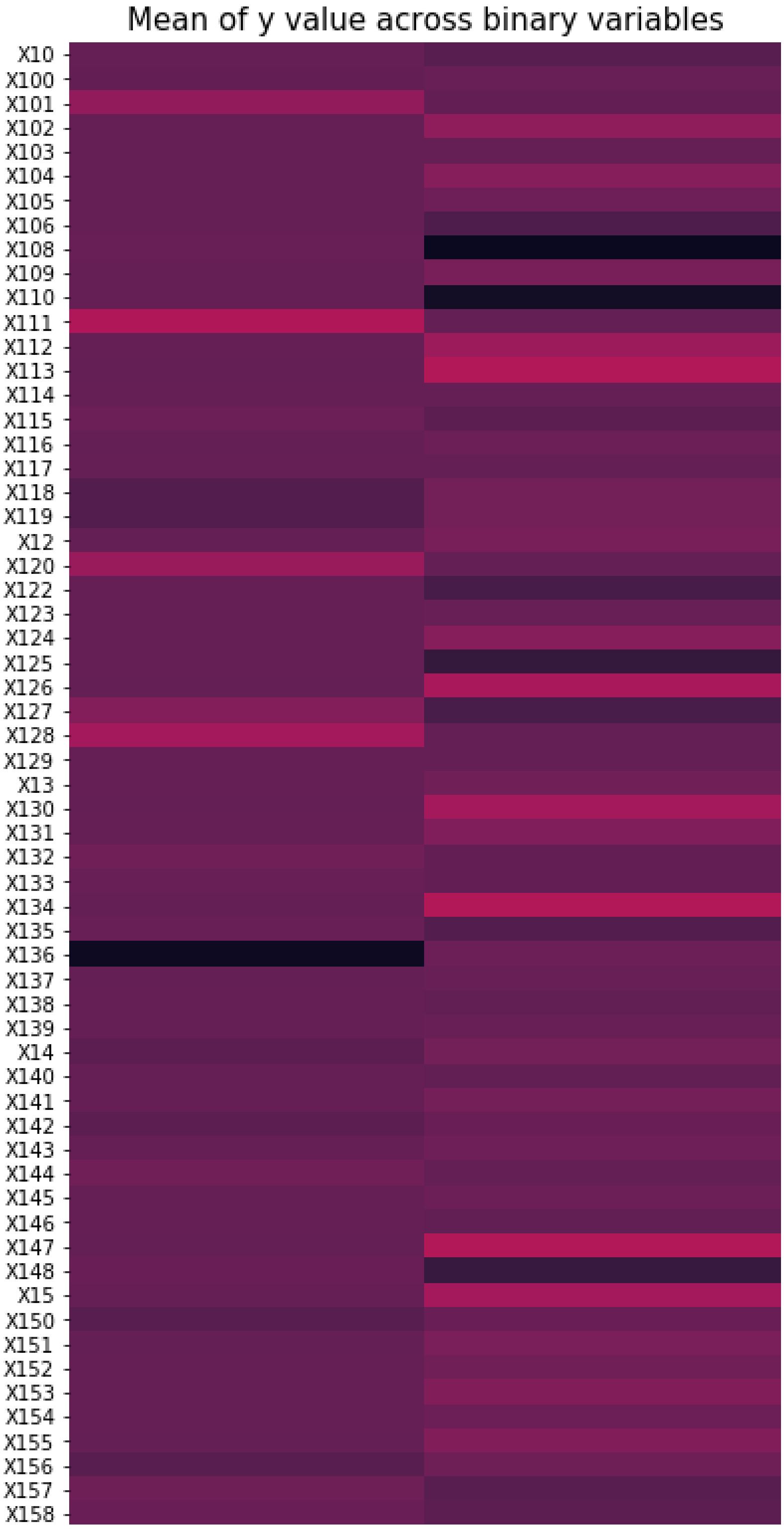
DISMISS

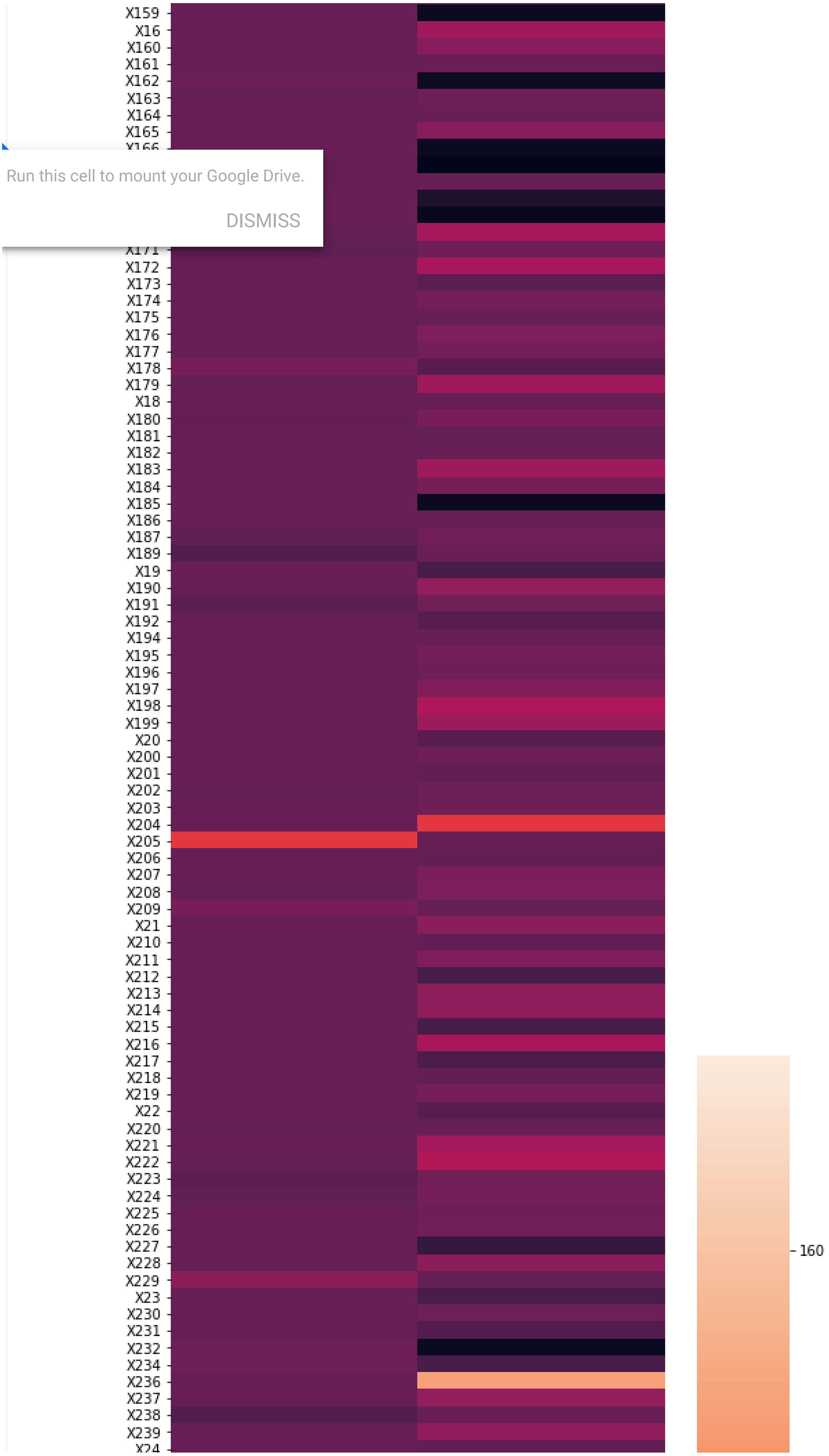
```
zero_mean_list = []
one_mean_list = []
cols_list = unique_values_dict['[0, 1]']
for col in cols_list:
    zero_mean_list.append(train_df.ix[train_df[col]==0].y.mean())
    one_mean_list.append(train_df.ix[train_df[col]==1].y.mean())

new_df = pd.DataFrame({"column_name":cols_list+cols_list, "value":[0]*len(cols_list) + [1
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()
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:5: DeprecationWar
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
                    a.org/pandas-docs/stable/indexing.html#ix-indexer-is-deprec

                    on3.6/dist-packages/ipykernel_launcher.py:6: DeprecationWar
                    Please use
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/indexing.html#ix-indexer-is-deprec
```

Run this cell to mount your Google Drive.

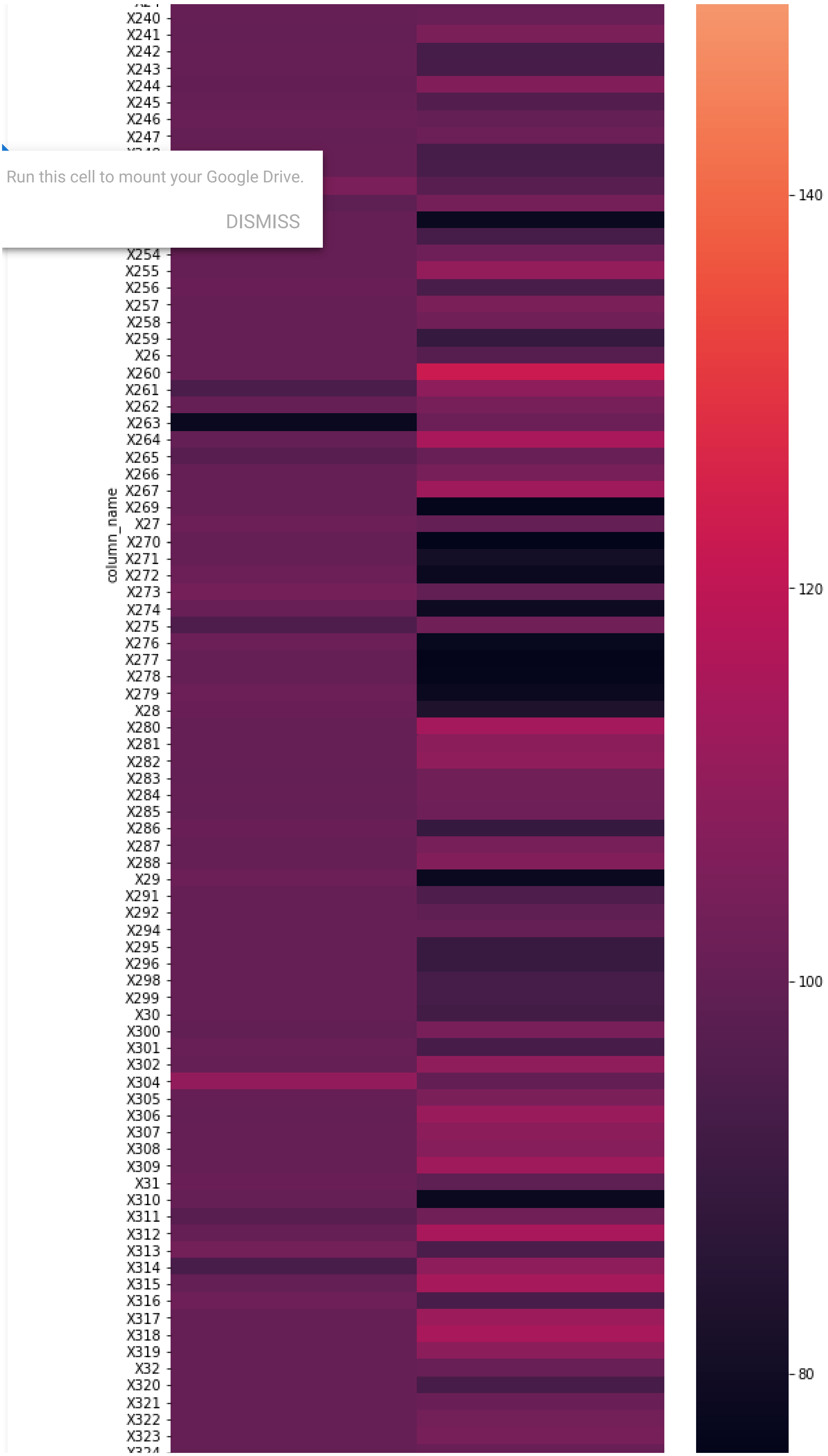
DISMISS





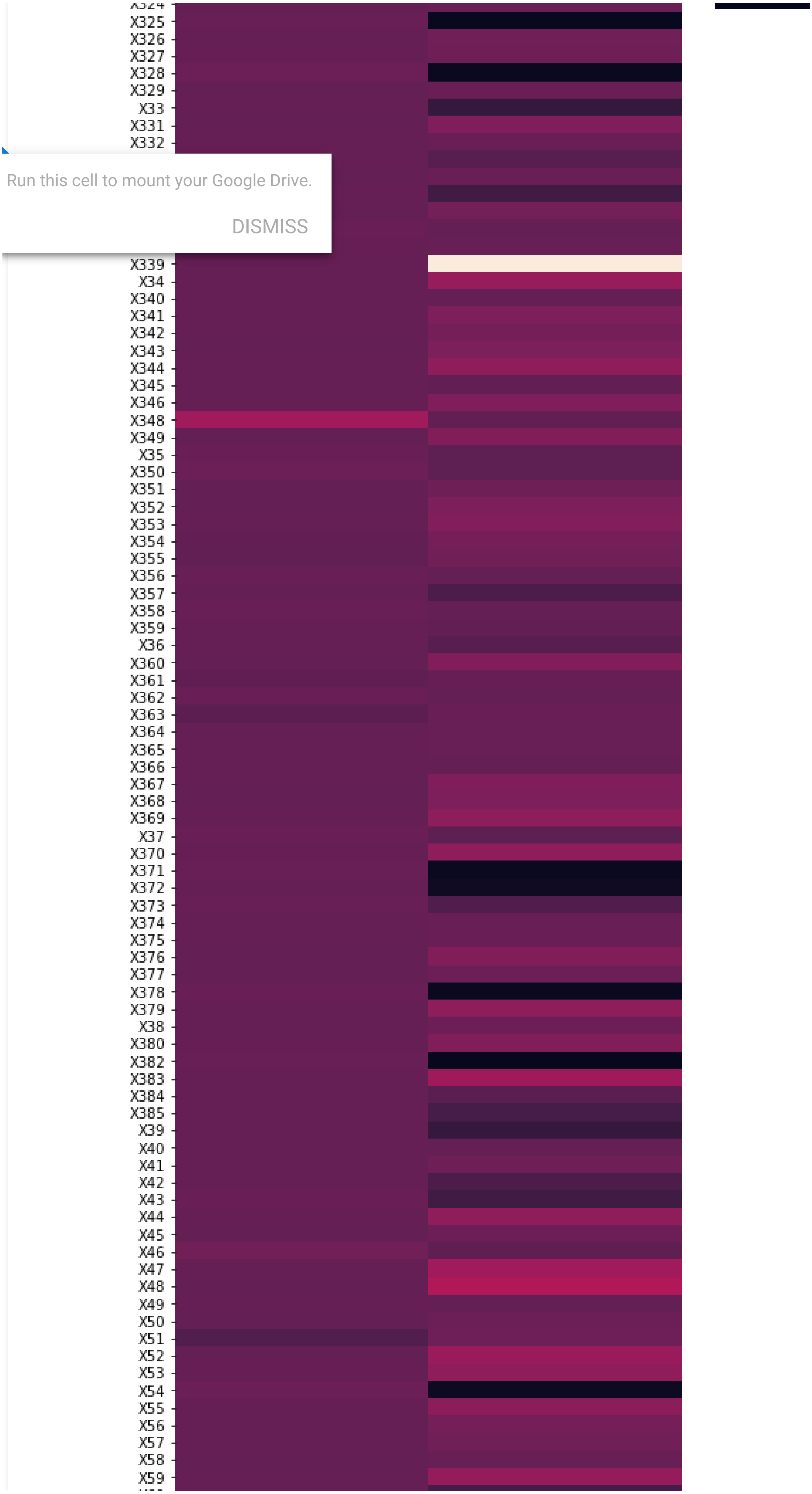
Run this cell to mount your Google Drive.

DISMISS

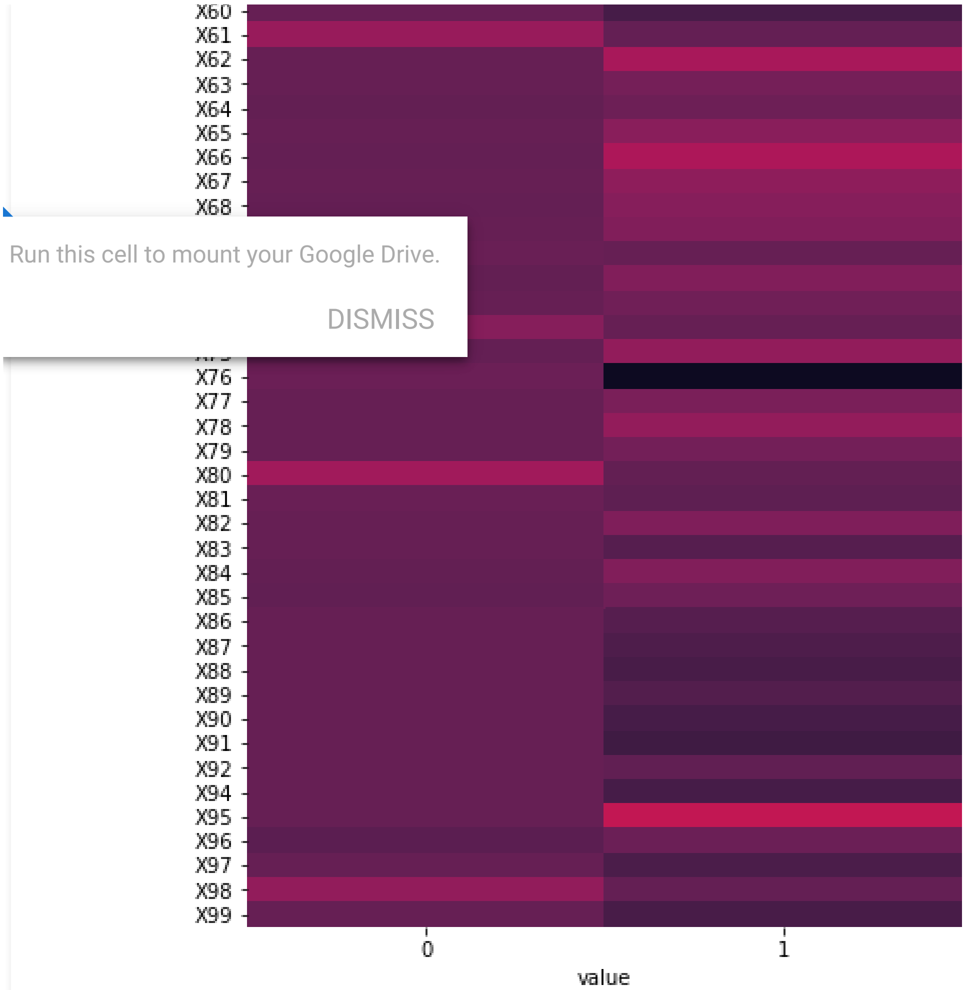


Run this cell to mount your Google Drive.

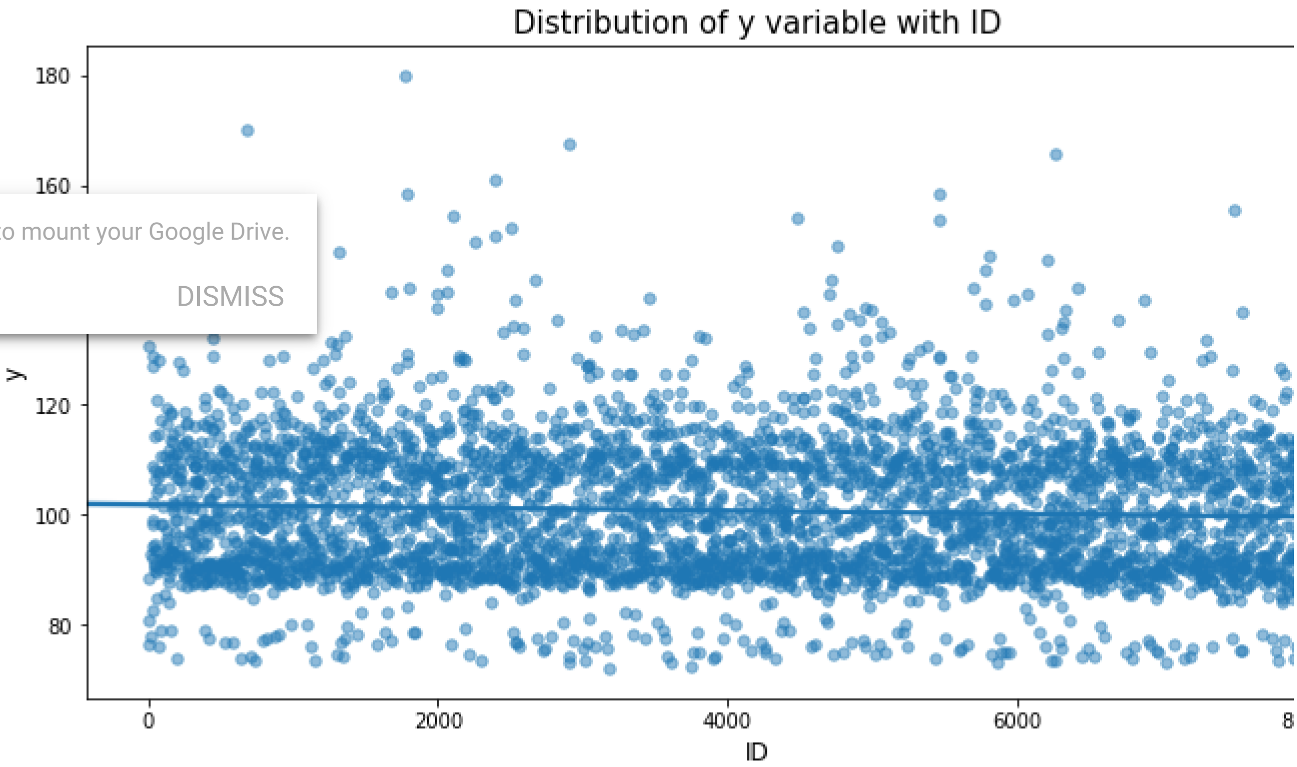
DISMISS



Run this cell to mount your Google Drive.



```
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()
```

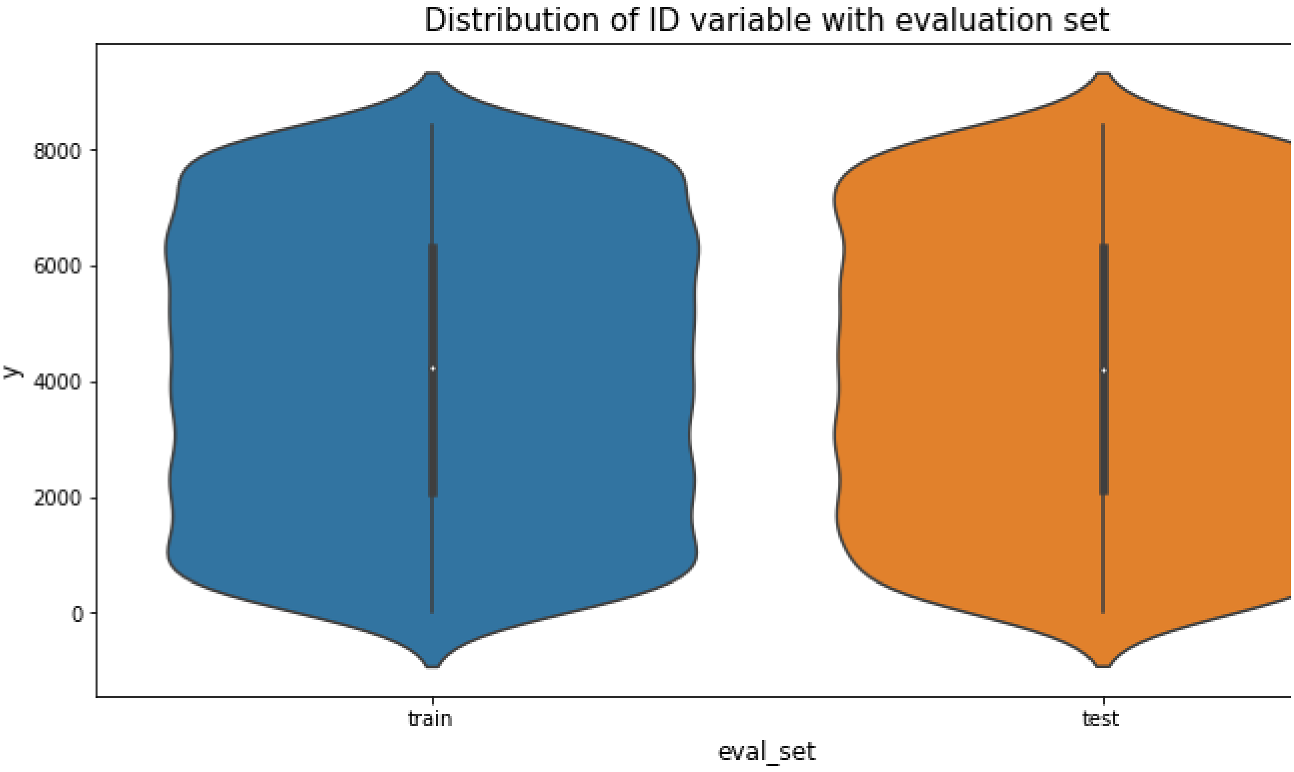


Run this cell to mount your Google Drive.

DISMISS

```
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)

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()
```

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



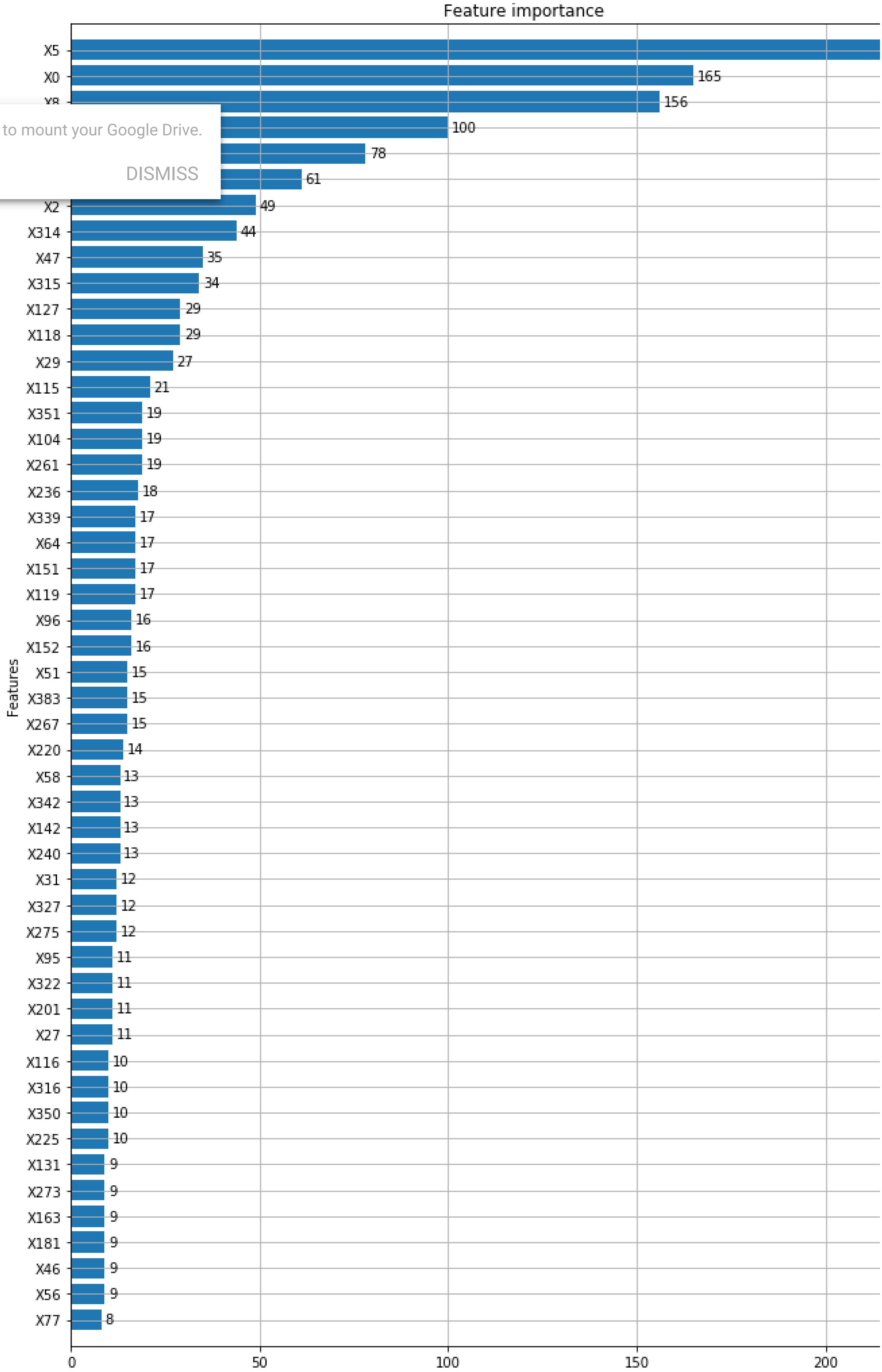
```
for f in ["X0", "X1", "X2", "X3", "X4", "X5", "X6", "X8"]:
        lbl = preprocessing.LabelEncoder()
        lbl.fit(list(train_df[f].values))
        train_df[f] = lbl.transform(list(train_df[f].values))

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

Run this cell to mount your Google Drive.

DISMISS

```
                          s #
                          rain):
                          bel()
                          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_s

# 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()
```

```
[06:18:03] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear
```



Run this cell to mount your Google Drive.

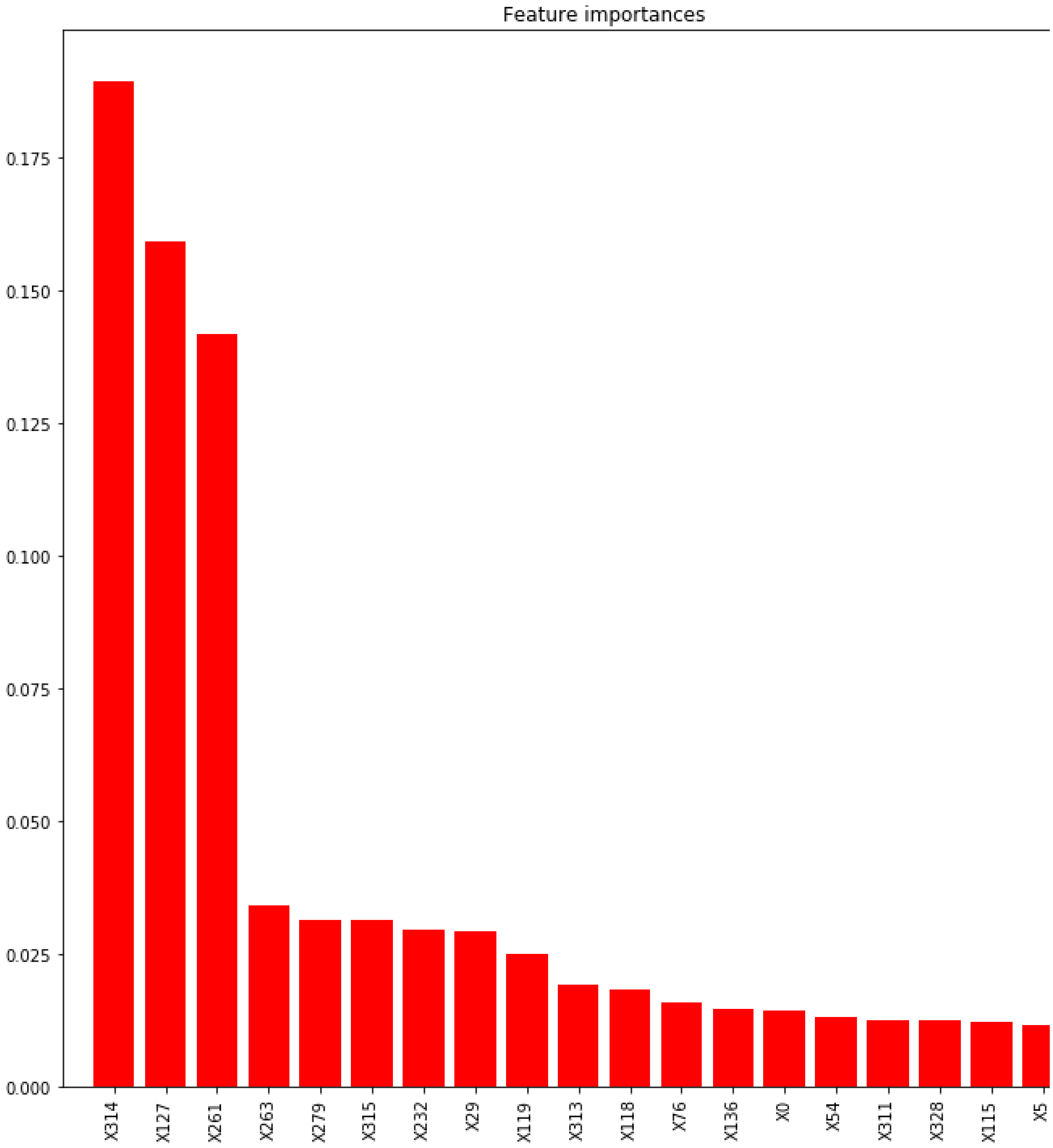
DISMISS

F score

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


                                   e_importances_
                                   _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()
```

Run this cell to mount your Google Drive.

DISMISS



Run this cell to mount your Google Drive.

DISMISS