Imports and setup

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import RANSACRegressor
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score

import scipy as sp
from scipy import stats


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

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.mou
```

Load and proceses data

Load data from csv

```
path_after_2010 = '/content/gdrive/My Drive/NIOSH Project/data/violations_processed_metal.csv
violation_data = pd.read_csv(path_after_2010)
```

Split into features and targets

```
FEATURES = ['MINE_TYPE', 'COAL_METAL_IND', 'SIG_SUB', 'LIKELIHOOD',
            'INJ_ILLNESS', 'NO_AFFECTED', 'NEGLIGENCE', 'VIOLATOR_VIOLATION_CNT',
            'VIOLATOR_INSPECTION_DAY_CNT']
TARGETS = ['PROPOSED_PENALTY']

X = violation_data[FEATURES]
y = violation_data[TARGETS]
```

Encode and scale features

```
# Define which columns should be encoded vs scaled
columns_to_encode = ['MINE_TYPE', 'COAL_METAL_IND', 'LIKELIHOOD', 'INJ_ILLNESS', 'SIG_SUB', '
columns_to_scale  = ['VIOLATOR_VIOLATION_CNT', 'NO_AFFECTED', 'VIOLATOR_INSPECTION_DAY_CNT']

# Instantiate encoder/scaler
scaler = StandardScaler()
ohe = OneHotEncoder(sparse=False)

# Scale and Encode Separate Columns
scaled_columns  = scaler.fit_transform(X[columns_to_scale])
encoded_columns = ohe.fit_transform(X[columns_to_encode])

# Concatenate (Column-Bind) Processed Columns Back Together
X_pre = np.concatenate([scaled_columns, encoded_columns], axis=1)
np.nan_to_num(X_pre, copy=False)

print('Features shape:', X_pre.shape)
```

```
Features shape: (1302293, 23)
```

Split data into training and testing datasets

```
X_train, X_test, y_train, y_test = train_test_split(X_pre, y, test_size = 0.25, random_state

print('X_train shape:', X_train.shape)
print('X_test shape:', X_test.shape)
print('y_train shape:', y_train.shape)
print('y_train shape:', y_test.shape)
```

```
X_train shape: (976719, 23)
X_test shape: (325574, 23)
y_train shape: (976719, 1)
y_train shape: (325574, 1)
```

Define and Fit Simple Linear Regression Model

```
lin_reg = LinearRegression().fit(X_train, y_train)
```

```
lin_reg_y_pred_train = lin_reg.predict(X_train)
lin_reg_y_pred_test = lin_reg.predict(X_test)

lin_reg_r_squared_train = r2_score(y_train, lin_reg_y_pred_train)
lin_reg_r_squared_test = r2_score(y_test, lin_reg_y_pred_test)
```

```
lin_reg_r_squared_test = r2_score(y_test, lin_reg_y_pred_test)

lin_reg_mae_train = mean_absolute_error(y_train, lin_reg_y_pred_train)
lin_reg_mae_test = mean_absolute_error(y_test, lin_reg_y_pred_test)

print('Simple Linear Regression Scores:')
print('Training R^2 = %.3f' % lin_reg_r_squared_train)
print('Training MAE = %.3f' % lin_reg_mae_train)

print('\nTesting R^2 = %.3f' % lin_reg_r_squared_test)
print('Testing MAE = %.3f' % lin_reg_mae_test)
```

```
Simple Linear Regression Scores:
Training R^2 = 0.207
Training MAE = 493.394

Testing R^2 = 0.221
Testing MAE = 491.366
```

Define and Fit RANSAC Regressor

```
ransac_reg = RANSACRegressor().fit(X_train, y_train)
```

```
ransac_reg_y_pred_train = ransac_reg.predict(X_train)
ransac_reg_y_pred_test = ransac_reg.predict(X_test)

ransac_reg_r_squared_train = r2_score(y_train, ransac_reg_y_pred_train)
ransac_reg_r_squared_test = r2_score(y_test, ransac_reg_y_pred_test)

ransac_reg_mae_train = mean_absolute_error(y_train, ransac_reg_y_pred_train)
ransac_reg_mae_test = mean_absolute_error(y_test, ransac_reg_y_pred_test)

print('RANSAC Regression Scores:')
print('Training R^2 = %.3f' % ransac_reg_r_squared_train)
print('Training MAE = %.3f' % ransac_reg_mae_train)

print('\nTesting R^2 = %.3f' % ransac_reg_r_squared_test)
print('Testing MAE = %.3f' % ransac_reg_mae_test)
```

```
RANSAC Regression Scores:
Training R^2 = -6.513
Training MAE = 773.190

Testing R^2 = -8.926
Testing MAE = 770.805
```

Removing Outliers in Target Data

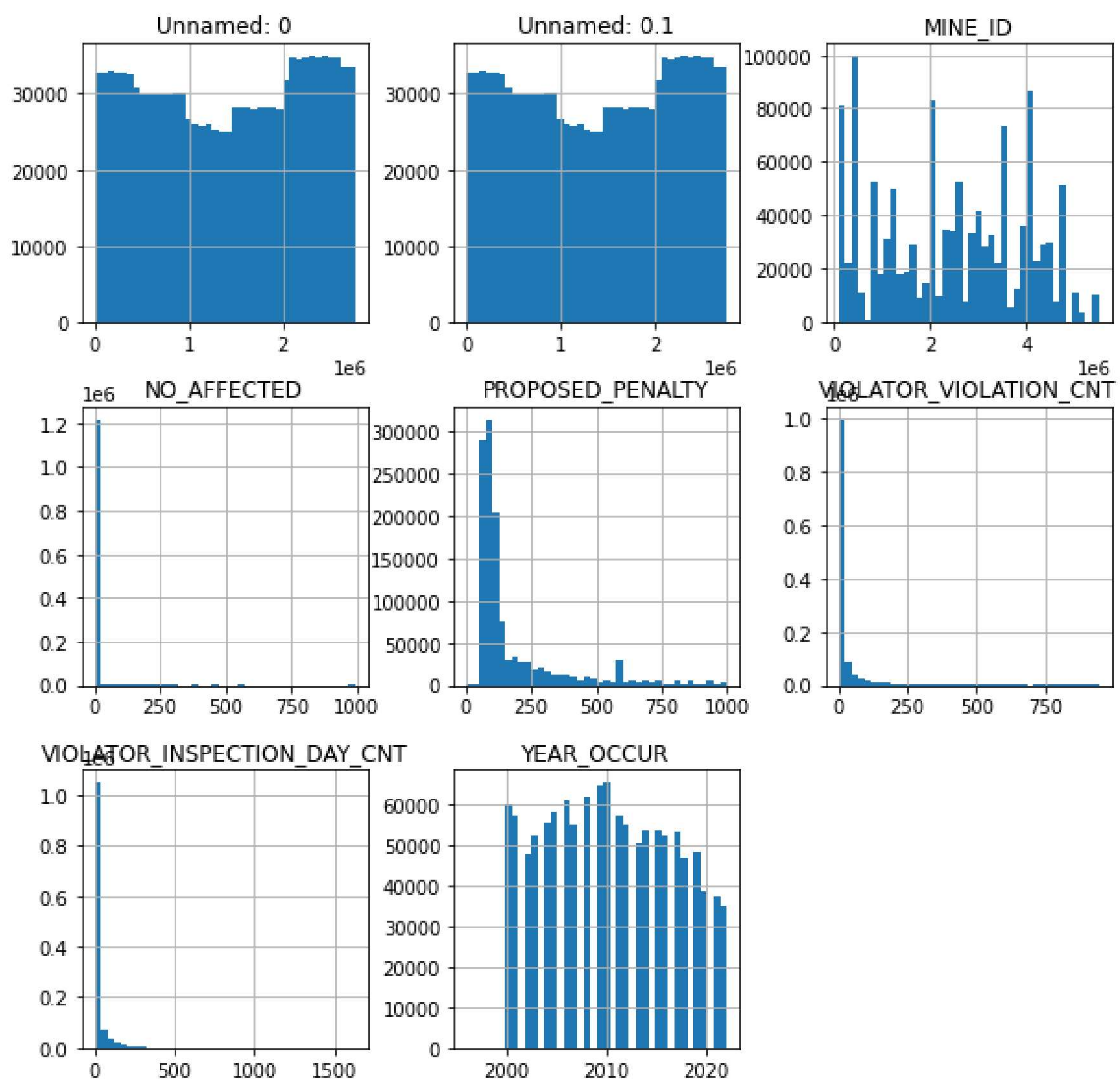
Most of the data in the PROPOSED_PENALTY column seems to be less than 500. So, let's try removing large values from the dataset.

```
MAX_VALUE = 1000
violations_cliped = violation_data[violation_data['PROPOSED_PENALTY'] <= MAX_VALUE]

print('Total samples = %d' % len(violation_data))
print('Total samples with PROPOSED_PENALTY <= %d: %d' % (MAX_VALUE, len(violations_cliped)))
```

```
Total samples = 1302293
Total samples with PROPOSED_PENALTY <= 1000: 1216618
```

```
hist = violations_cliped.hist(figsize=(10,10), bins=40)
```



Prepare Inlier Data

```
X_inliers = violations_cliped[FEATURES]
y_inliers = violations_cliped[TARGETS]

# Instantiate encoder/scaler
scaler = StandardScaler()
ohe = OneHotEncoder(sparse=False)

# Scale and Encode Separate Columns
scaled_columns_inliers  = scaler.fit_transform(X_inliers[columns_to_scale])
encoded_columns_inliers = ohe.fit_transform(X_inliers[columns_to_encode])

# Concatenate (Column-Bind) Processed Columns Back Together
X_pre_inliers = np.concatenate([scaled_columns_inliers, encoded_columns_inliers], axis=1)
np.nan_to_num(X_pre_inliers, copy=False)

X_train_inliers, X_test_inliers, y_train_inliers, y_test_inliers = train_test_split(X_pre_inl
```

Inlier Linear Regression

```
lin_reg_inliers = LinearRegression().fit(X_train_inliers, y_train_inliers)
```

```
lin_reg_inliers_y_pred_train = lin_reg_inliers.predict(X_train_inliers)
lin_reg_inliers_y_pred_test = lin_reg_inliers.predict(X_test_inliers)

lin_reg_inliers_r_squared_train = r2_score(y_train_inliers, lin_reg_inliers_y_pred_train)
lin_reg_inliers_r_squared_test = r2_score(y_test_inliers, lin_reg_inliers_y_pred_test)

lin_reg_inliers_mae_train = mean_absolute_error(y_train_inliers, lin_reg_inliers_y_pred_train
lin_reg_inliers_mae_test = mean_absolute_error(y_test_inliers, lin_reg_inliers_y_pred_test)

print('Inlier Simple Linear Regression Scores:')
print('Training R^2 = %.3f' % lin_reg_inliers_r_squared_train)
print('Training MAE = %.3f' % lin_reg_inliers_mae_train)

print('\nTesting R^2 = %.3f' % lin_reg_inliers_r_squared_test)
print('Testing MAE = %.3f' % lin_reg_inliers_mae_test)
```

```
Inlier Simple Linear Regression Scores:
Training R^2 = 0.334
Training MAE = 86.944

Testing R^2 = 0.334
Testing MAE = 86.653
```

Inlier RANSAC Regression

```
ransac_reg_inliers = RANSACRegressor(random_state=35).fit(X_train_inliers, y_train_inliers)


ransac_reg_inliers_y_pred_train = ransac_reg_inliers.predict(X_train_inliers)
ransac_reg_inliers_y_pred_test = ransac_reg_inliers.predict(X_test_inliers)

ransac_reg_inliers_r_squared_train = r2_score(y_train_inliers, ransac_reg_inliers_y_pred_trai
ransac_reg_inliers_r_squared_test = r2_score(y_test_inliers, ransac_reg_inliers_y_pred_test)

ransac_reg_inliers_mae_train = mean_absolute_error(y_train_inliers, ransac_reg_inliers_y_pred_
ransac_reg_inliers_mae_test = mean_absolute_error(y_test_inliers, ransac_reg_inliers_y_pred_t

print('Inlier RANSAC Regression Scores:')
print('Training R^2 = %.3f' % ransac_reg_inliers_r_squared_train)
print('Training MAE = %.3f' % ransac_reg_inliers_mae_train)

print('\nTesting R^2 = %.3f' % ransac_reg_inliers_r_squared_test)
print('Testing MAE = %.3f' % ransac_reg_inliers_mae_test)
```

```
Inlier RANSAC Regression Scores:
Training R^2 = 0.139
Training MAE = 87.144

Testing R^2 = 0.141
Testing MAE = 86.701
```

Focusing on Outliers

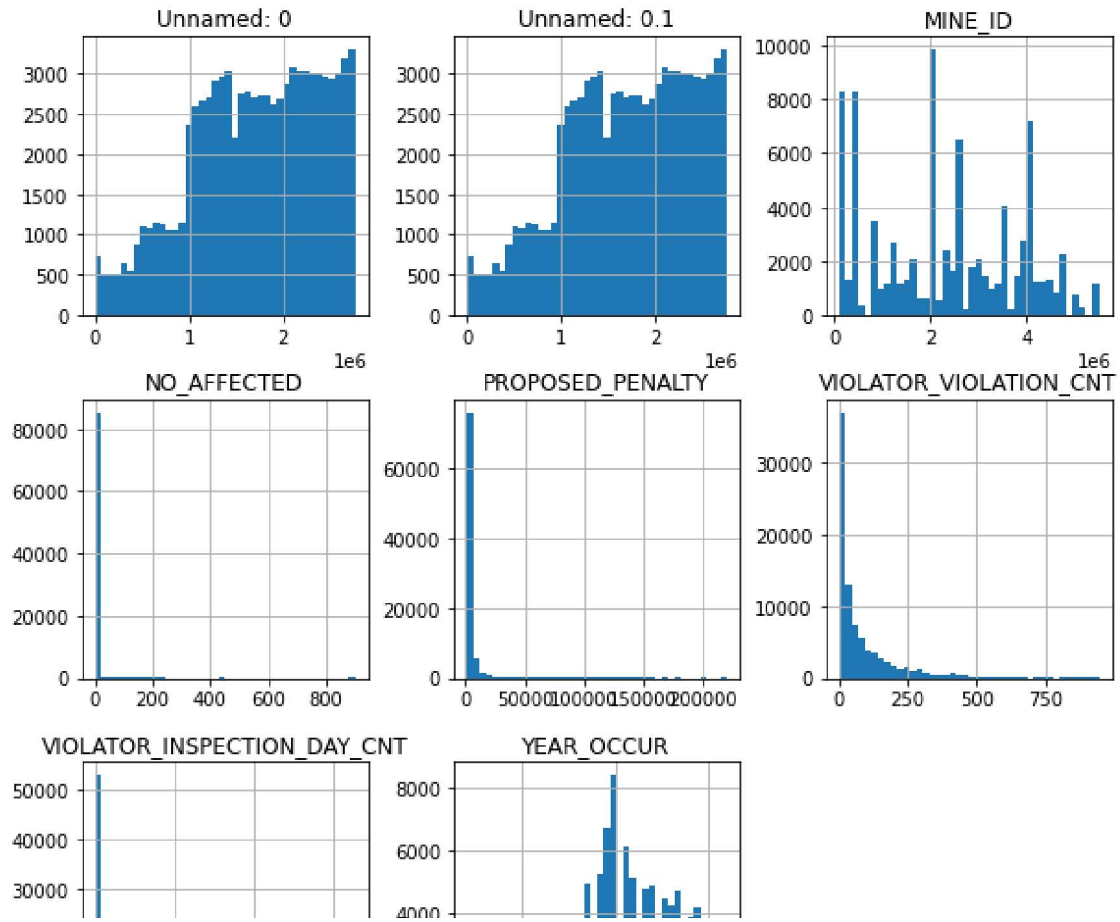
Now lets see if we can fit a model to the larger values.

```
violations_outliers = violation_data[violation_data['PROPOSED_PENALTY'] > MAX_VALUE]

print('Total samples = %d' % len(violation_data))
print('Total samples with PROPOSED_PENALTY > %d: %d' % (MAX_VALUE, len(violations_outliers)))
```

```
Total samples = 1302293
Total samples with PROPOSED_PENALTY > 1000: 85675
```

```
hist = violations_outliers.hist(figsize=(10,10), bins=40)
```



Process outlier data

```
X_outliers = violations_outliers[FEATURES]
y_outliers = violations_outliers[TARGETS]

# Instantiate encoder/scaler
scaler = StandardScaler()
ohe = OneHotEncoder(sparse=False)

# Scale and Encode Separate Columns
scaled_columns_outliers  = scaler.fit_transform(X_outliers[columns_to_scale])
encoded_columns_outliers = ohe.fit_transform(X_outliers[columns_to_encode])

# Concatenate (Column-Bind) Processed Columns Back Together
X_pre_outliers = np.concatenate([scaled_columns_outliers, encoded_columns_outliers], axis=1)
np.nan_to_num(X_pre_outliers, copy=False)

X_train_outliers, X_test_outliers, y_train_outliers, y_test_outliers = train_test_split(X_pre
```

Outlier Linear Regression

```
lin_reg_outliers = LinearRegression().fit(X_train_outliers, y_train_outliers)

lin_reg_outliers_y_pred_train = lin_reg_outliers.predict(X_train_outliers)
lin_reg_outliers_y_pred_test = lin_reg_outliers.predict(X_test_outliers)

lin_reg_outliers_r_squared_train = r2_score(y_train_outliers, lin_reg_outliers_y_pred_train)
lin_reg_outliers_r_squared_test = r2_score(y_test_outliers, lin_reg_outliers_y_pred_test)

lin_reg_outliers_mae_train = mean_absolute_error(y_train_outliers, lin_reg_outliers_y_pred_tr
lin_reg_outliers_mae_test = mean_absolute_error(y_test_outliers, lin_reg_outliers_y_pred_test

print('Outlier Simple Linear Regression Scores:')
print('Training R^2 = %.3f' % lin_reg_outliers_r_squared_train)
print('Training MAE = %.3f' % lin_reg_outliers_mae_train)

print('\nTesting R^2 = %.3f' % lin_reg_outliers_r_squared_test)
print('Testing MAE = %.3f' % lin_reg_outliers_mae_test)
```

```
Outlier Simple Linear Regression Scores:
Training R^2 = 0.219
Training MAE = 2948.273

Testing R^2 = 0.227
Testing MAE = 2857.880
```

Outlier RANSAC Regression

```
ransac_reg_outliers = RANSACRegressor(random_state=2).fit(X_train_outliers, y_train_outliers)


ransac_reg_outliers_y_pred_train = ransac_reg_outliers.predict(X_train_outliers)
ransac_reg_outliers_y_pred_test = ransac_reg_outliers.predict(X_test_outliers)

ransac_reg_outliers_r_squared_train = r2_score(y_train_outliers, ransac_reg_outliers_y_pred_t
ransac_reg_outliers_r_squared_test = r2_score(y_test_outliers, ransac_reg_outliers_y_pred_tes

ransac_reg_outliers_mae_train = mean_absolute_error(y_train_outliers, ransac_reg_outliers_y_p
ransac_reg_outliers_mae_test = mean_absolute_error(y_test_outliers, ransac_reg_outliers_y_pre

print('Outlier RANSAC Regression Scores:')
print('Training R^2 = %.3f' % ransac_reg_outliers_r_squared_train)
print('Training MAE = %.3f' % ransac_reg_outliers_mae_train)

print('\nTesting R^2 = %.3f' % ransac_reg_outliers_r_squared_test)
print('Testing MAE = %.3f' % ransac_reg_outliers_mae_test)
```

```
Outlier RANSAC Regression Scores:
Training R^2 = 0.031
Training MAE = 2531.400
```

```
Testing R^2 = 0.043
Testing MAE = 2447.506
```