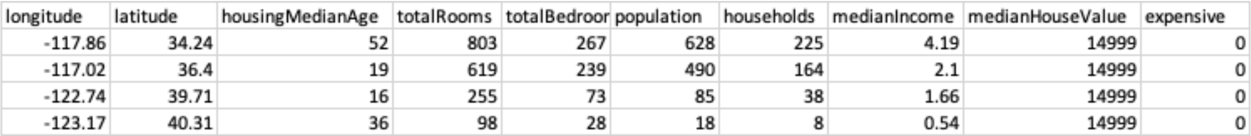
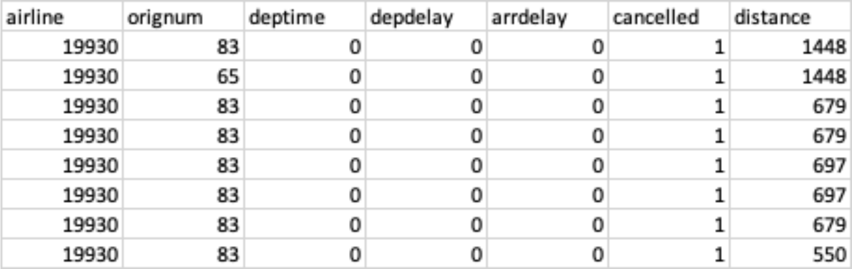
Sample Charts from ML Analysis of 3 Data Sets

Cal Housing



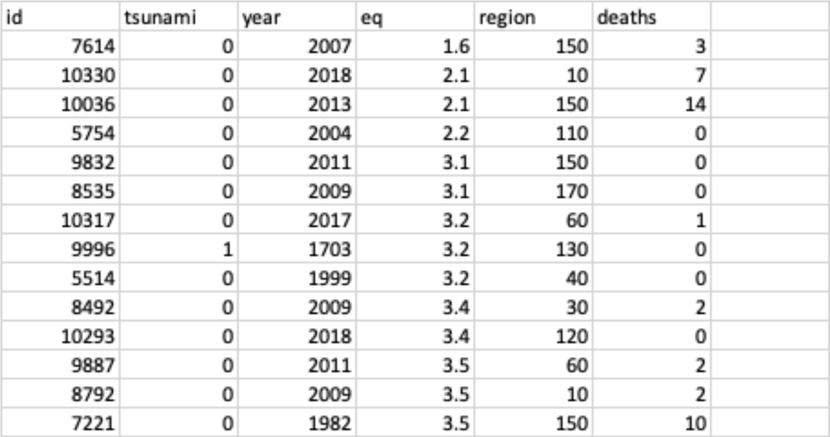
longitude	latitude	housingMedianAge	totalRooms	totalBedroor	population	households	medianIncome	medianHouseValue	expensive
-117.86	34.24	52	803	267	628	225	4.19	14999	0
-117.02	36.4	19	619	239	490	164	2.1	14999	0
-122.74	39.71	16	255	73	85	38	1.66	14999	0
-123.17	40.31	36	98	28	18	8	0.54	14999	0

Flights



airline	orignum	deptime	depdelay	arrdelay	cancelled	distance
19930	83	0	0	0	1	1448
19930	65	0	0	0	1	1448
19930	83	0	0	0	1	679
19930	83	0	0	0	1	679
19930	83	0	0	0	1	697
19930	83	0	0	0	1	697
19930	83	0	0	0	1	679
19930	83	0	0	0	1	550

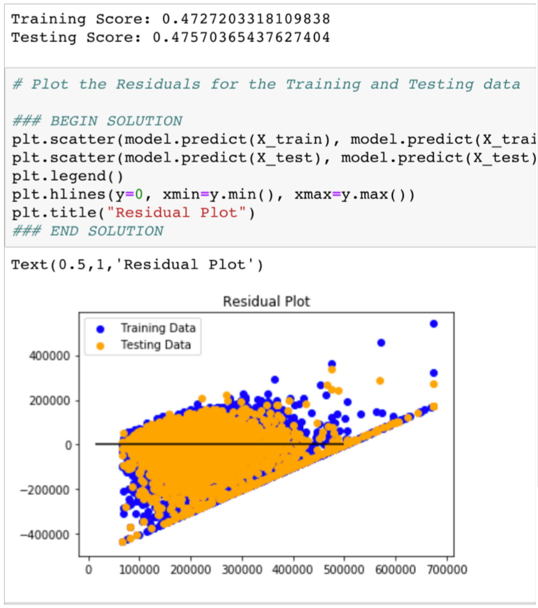
USGS Earthquakes



id	tsunami	year	eq	region	deaths	
7614	0	2007	1.6	150	3	
10330	0	2018	2.1	10	7	
10036	0	2013	2.1	150	14	
5754	0	2004	2.2	110	0	
9832	0	2011	3.1	150	0	
8535	0	2009	3.1	170	0	
10317	0	2017	3.2	60	1	
9996	1	1703	3.2	130	0	
5514	0	1999	3.2	40	0	
8492	0	2009	3.4	30	2	
10293	0	2018	3.4	120	0	
9887	0	2011	3.5	60	2	
8792	0	2009	3.5	10	2	
7221	0	1982	3.5	150	10	

Sample output from data sets and algorithms: BEER FOAM

Cal Housing



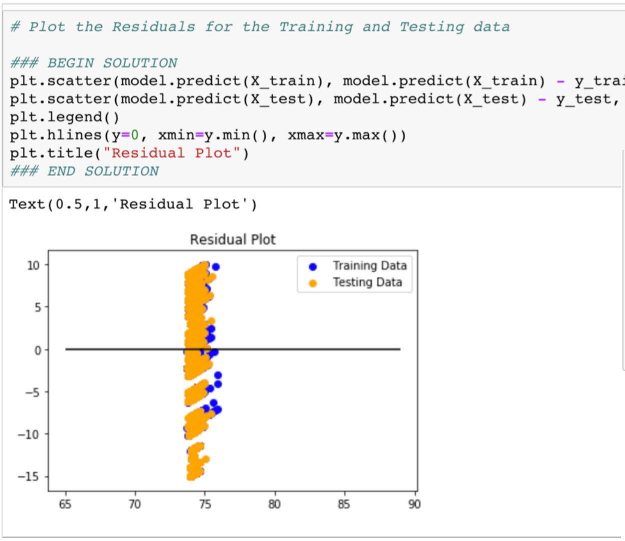
```
Training Score: 0.4727203318109838
Testing Score: 0.47570365437627404
```

```
# Plot the Residuals for the Training and Testing data

### BEGIN SOLUTION
plt.scatter(model.predict(X_train), model.predict(X_trai
plt.scatter(model.predict(X_test), model.predict(X_test)
plt.legend()
plt.hlines(y=0, xmin=y.min(), xmax=y.max())
plt.title("Residual Plot")
### END SOLUTION
```

```
Text(0.5,1,'Residual Plot')
```

Flights

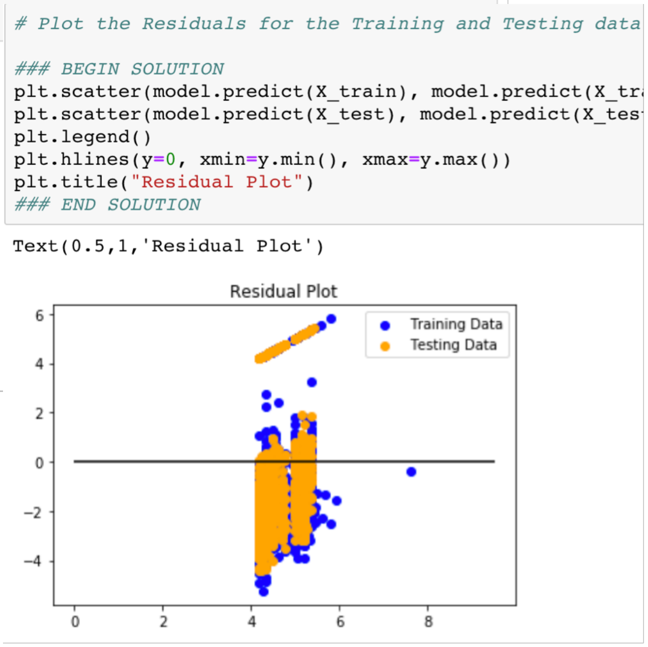


```
# Plot the Residuals for the Training and Testing data

### BEGIN SOLUTION
plt.scatter(model.predict(X_train), model.predict(X_train) - y_train, c="blue", label="Training Data")
plt.scatter(model.predict(X_test), model.predict(X_test) - y_test, c="orange", label="Testing Data")
plt.legend()
plt.hlines(y=0, xmin=y.min(), xmax=y.max())
plt.title("Residual Plot")
### END SOLUTION
```

```
Text(0.5,1,'Residual Plot')
```

USGS Earthquakes



```
# Plot the Residuals for the Training and Testing data

### BEGIN SOLUTION
plt.scatter(model.predict(X_train), model.predict(X_tra
plt.scatter(model.predict(X_test), model.predict(X_tes
plt.legend()
plt.hlines(y=0, xmin=y.min(), xmax=y.max())
plt.title("Residual Plot")
### END SOLUTION
```

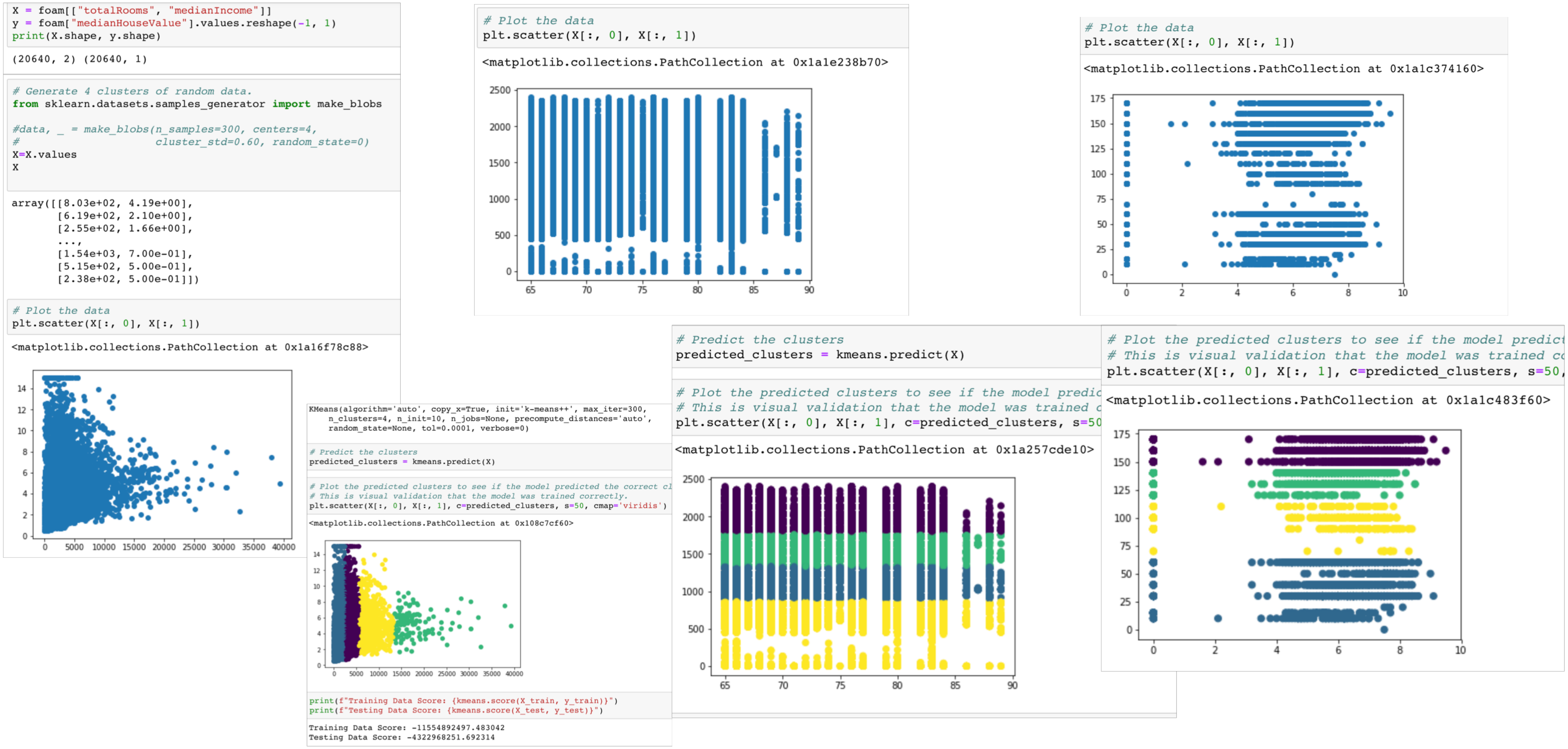
```
Text(0.5,1,'Residual Plot')
```

Sample output from data sets and algorithms: K-Means

Cal Housing

Flights

USGS Earthquakes



```
X = foam[["totalRooms", "medianIncome"]]
y = foam["medianHouseValue"].values.reshape(-1, 1)
print(X.shape, y.shape)
```

```
(20640, 2) (20640, 1)
```

```
# Generate 4 clusters of random data.
from sklearn.datasets.samples_generator import make_blobs

#data, _ = make_blobs(n_samples=300, centers=4,
#                     cluster_std=0.60, random_state=0)
X=X.values
X
```

```
array([[8.03e+02, 4.19e+00],
       [6.19e+02, 2.10e+00],
       [2.55e+02, 1.66e+00],
       ...,
       [1.54e+03, 7.00e-01],
       [5.15e+02, 5.00e-01],
       [2.38e+02, 5.00e-01]])
```

```
# Plot the data
plt.scatter(X[:, 0], X[:, 1])
```

```
<matplotlib.collections.PathCollection at 0x1a16f78c88>
```

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=4, n_init=10, n_jobs=None, precompute_distances='auto',
    random_state=None, tol=0.0001, verbose=0)
```

```
# Predict the clusters
predicted_clusters = kmeans.predict(X)
```

```
# Plot the predicted clusters to see if the model predicted the correct cl
# This is visual validation that the model was trained correctly.
plt.scatter(X[:, 0], X[:, 1], c=predicted_clusters, s=50, cmap='viridis')
```

```
<matplotlib.collections.PathCollection at 0x108c7cf60>
```

```
print(f"Training Data Score: {kmeans.score(X_train, y_train)}")
print(f"Testing Data Score: {kmeans.score(X_test, y_test)}")
```

```
Training Data Score: -11554892497.483042
Testing Data Score: -4322968251.692314
```

```
# Plot the data
plt.scatter(X[:, 0], X[:, 1])
```

```
<matplotlib.collections.PathCollection at 0x1a1e238b70>
```

```
# Predict the clusters
predicted_clusters = kmeans.predict(X)
```

```
# Plot the predicted clusters to see if the model predic
# This is visual validation that the model was trained c
plt.scatter(X[:, 0], X[:, 1], c=predicted_clusters, s=50
```

```
<matplotlib.collections.PathCollection at 0x1a257cde10>
```

```
# Plot the data
plt.scatter(X[:, 0], X[:, 1])
```

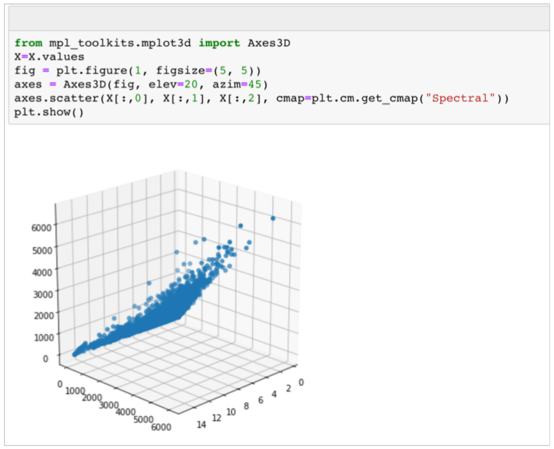
```
<matplotlib.collections.PathCollection at 0x1a1c374160>
```

```
# Plot the predicted clusters to see if the model predict
# This is visual validation that the model was trained co
plt.scatter(X[:, 0], X[:, 1], c=predicted_clusters, s=50,
```

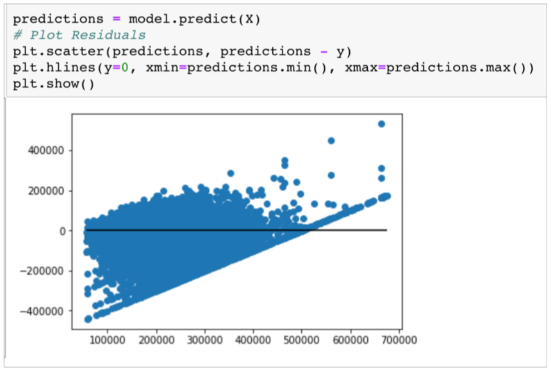
```
<matplotlib.collections.PathCollection at 0x1a1c483f60>
```

Sample output from data sets and algorithms: Multivariate Linear and Elastic

Cal Housing

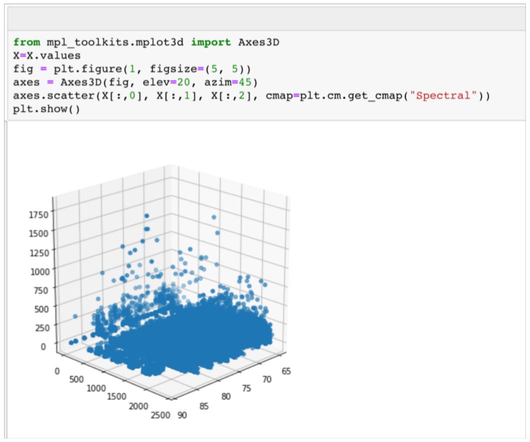


```
from mpl_toolkits.mplot3d import Axes3D
X=X.values
fig = plt.figure(1, figsize=(5, 5))
axes = Axes3D(fig, elev=20, azim=45)
axes.scatter(X[:,0], X[:,1], X[:,2], cmap=plt.cm.get_cmap("Spectral"))
plt.show()
```

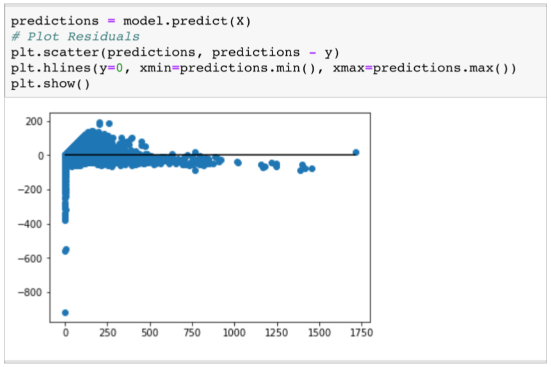


```
predictions = model.predict(X)
# Plot Residuals
plt.scatter(predictions, predictions - y)
plt.hlines(y=0, xmin=predictions.min(), xmax=predictions.max())
plt.show()
```

Flights

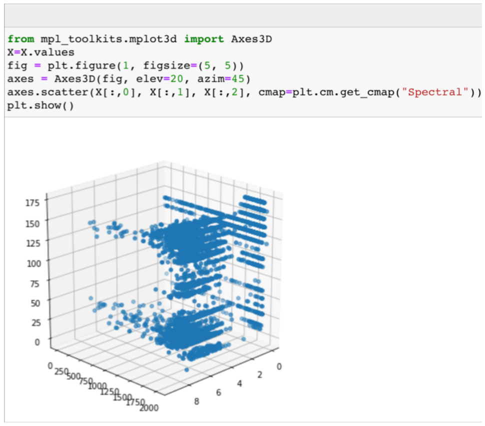


```
from mpl_toolkits.mplot3d import Axes3D
X=X.values
fig = plt.figure(1, figsize=(5, 5))
axes = Axes3D(fig, elev=20, azim=45)
axes.scatter(X[:,0], X[:,1], X[:,2], cmap=plt.cm.get_cmap("Spectral"))
plt.show()
```

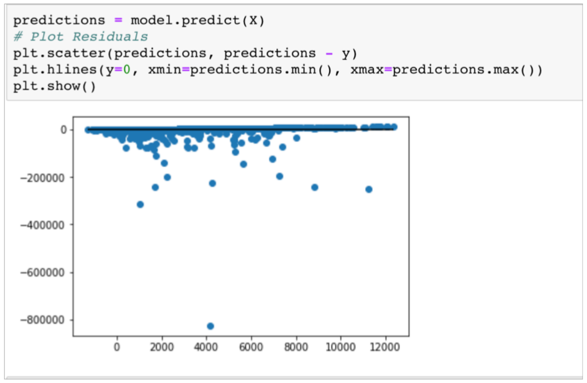


```
predictions = model.predict(X)
# Plot Residuals
plt.scatter(predictions, predictions - y)
plt.hlines(y=0, xmin=predictions.min(), xmax=predictions.max())
plt.show()
```

USGS Earthquakes



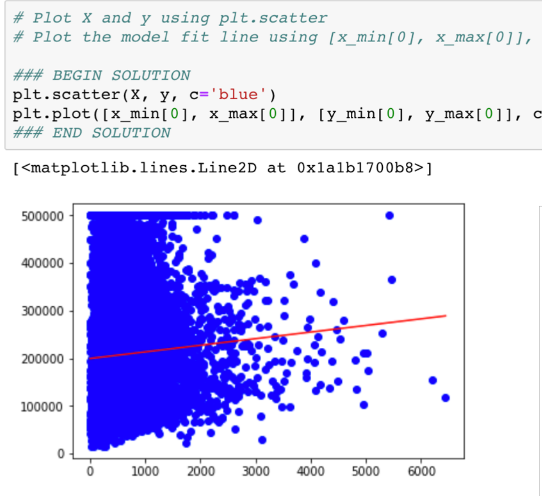
```
from mpl_toolkits.mplot3d import Axes3D
X=X.values
fig = plt.figure(1, figsize=(5, 5))
axes = Axes3D(fig, elev=20, azim=45)
axes.scatter(X[:,0], X[:,1], X[:,2], cmap=plt.cm.get_cmap("Spectral"))
plt.show()
```



```
predictions = model.predict(X)
# Plot Residuals
plt.scatter(predictions, predictions - y)
plt.hlines(y=0, xmin=predictions.min(), xmax=predictions.max())
plt.show()
```

Sample output from data sets and algorithms: Univariate

Cal Housing

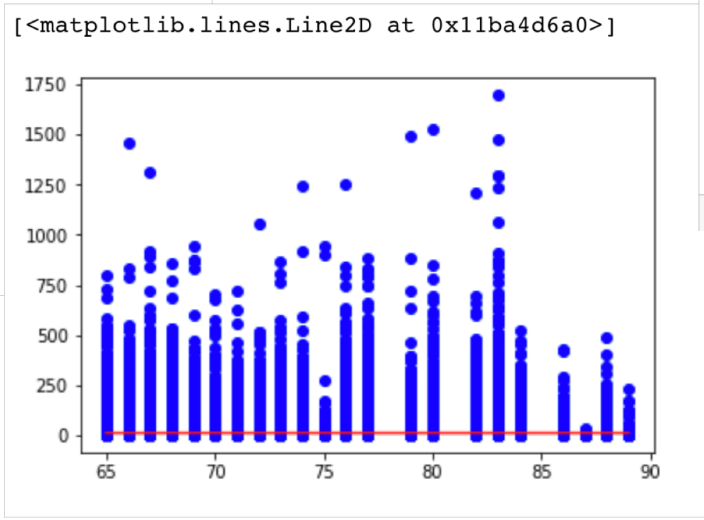


```
# Plot X and y using plt.scatter
# Plot the model fit line using [x_min[0], x_max[0]], [y_min[0], y_max[0]]

### BEGIN SOLUTION
plt.scatter(X, y, c='blue')
plt.plot([x_min[0], x_max[0]], [y_min[0], y_max[0]], c='red')
### END SOLUTION
```

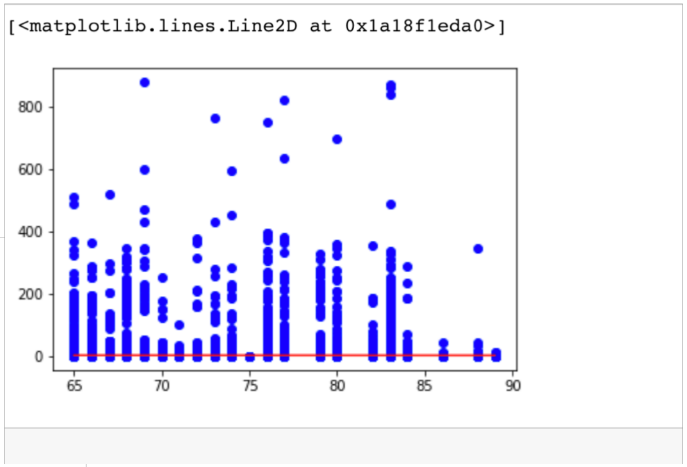
```
[<matplotlib.lines.Line2D at 0x1a1b1700b8>]
```

Flights



```
[<matplotlib.lines.Line2D at 0x11ba4d6a0>]
```

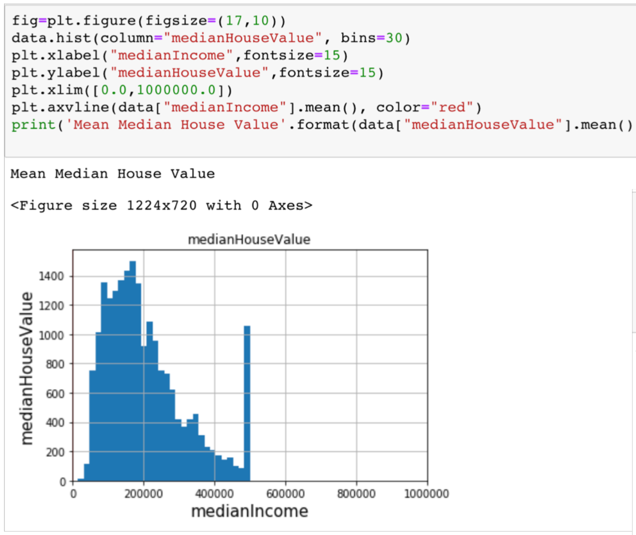
USGS Earthquakes



```
[<matplotlib.lines.Line2D at 0x1a18f1eda0>]
```

Sample output from data sets and algorithms: Simple Plot

Cal Housing

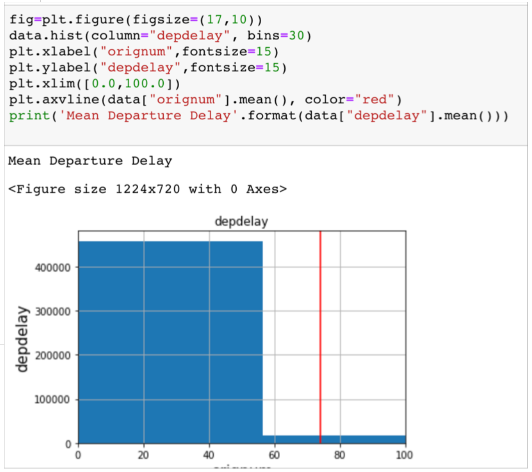


```
fig=plt.figure(figsize=(17,10))
data.hist(column="medianHouseValue", bins=30)
plt.xlabel("medianIncome",fontsize=15)
plt.ylabel("medianHouseValue",fontsize=15)
plt.xlim([0.0,1000000.0])
plt.axvline(data["medianIncome"].mean(), color="red")
print('Mean Median House Value'.format(data["medianHouseValue"].mean()))
```

```
Mean Median House Value

<Figure size 1224x720 with 0 Axes>
```

Flights

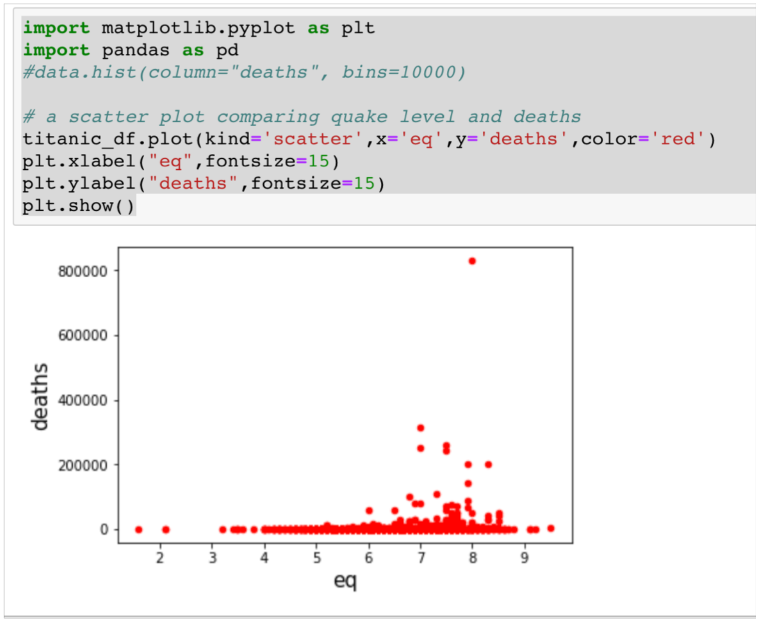


```
fig=plt.figure(figsize=(17,10))
data.hist(column="depdelay", bins=30)
plt.xlabel("orignum",fontsize=15)
plt.ylabel("depdelay",fontsize=15)
plt.xlim([0.0,100.0])
plt.axvline(data["orignum"].mean(), color="red")
print('Mean Departure Delay'.format(data["depdelay"].mean()))
```

```
Mean Departure Delay

<Figure size 1224x720 with 0 Axes>
```

USGS Earthquakes



```
import matplotlib.pyplot as plt
import pandas as pd
#data.hist(column="deaths", bins=10000)

# a scatter plot comparing quake level and deaths
titanic_df.plot(kind='scatter',x='eq',y='deaths',color='red')
plt.xlabel("eq",fontsize=15)
plt.ylabel("deaths",fontsize=15)
plt.show()
```