In [ ]:
```
File to explore the methodology behind a flood alert system.
The code is part of a hackathon event put on by Hackworks and MeteoHack.
It gives guidelines about how to use 3 categories of alerts (low, medium, high) by developing
a flood index.
Analysis was limited to watersheds/stations in Ontario

Analysis for Team GeoHack.
```

In [498]:
```
from math import sqrt
from numpy import concatenate
from matplotlib import pyplot
import pandas as pd
import json
import csv
import requests
import regex
#import urllib
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import mean_squared_error
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
import torch
import fastai
from fastai import *
from fastai.text import *
import plotnine
import pandasql

# Bokeh packages
from bokeh.io import output_notebook, output_file, show
from bokeh.plotting import figure
from bokeh.layouts import widgetbox
from bokeh.models import ColumnDataSource, HoverTool, Panel
from bokeh.models.widgets import DataTable, DateFormatter, TableColumn, HTMLTemplateFormatter
from bokeh.transform import factor_cmap
from bokeh.palettes import Paired12

output_notebook()
```

(https://bokeh.pydata.org) Loading BokehJS ...

Compile csv data into one dataframe

In [31]:
```
os.chdir("data")
```

In [32]:
```
filepath = "data"
```

Get an idea about the number of watersheds in Ontario

In [33]:
```
watersheds = pd.read_csv("../ON_hourly_hydrometric_June_July_todate.csv", encoding = "latin")
```

In [34]:
```
%%time
# Rename columns by stripping whitespace
ws_cols = []
for colnum in range(len(watersheds.columns)):
    ws_cols.append(watersheds.columns[colnum].lstrip())

watersheds.rename(columns=dict(zip(watersheds.columns,ws_cols)), \
                  inplace = True)
```

```
Wall time: 28.7 ms
```

In [35]:
```
%%time
# Number of watersheds in Ontario
num_watersheds = watersheds.groupby(by = "ID").size().reset_index(name = "count").shape
print("Number of watersheds in Ontario =", num_watersheds)
```

```
Number of watersheds in Ontario = (522, 2)
Wall time: 39.6 ms
```

Sample mean daily flow file

In [36]:
```
hydro_df = pd.read_csv("hydro_daily_mean_39.csv", encoding = "latin")
```

Number of stations - might not use

In [37]:
```
hydro_df.groupby(by = "STATION_NUMBER").ngroups
```

Out[37]:
```
27
```

Estimate the average number of records per station

In [494]:
```
hydro_df.groupby(by = "STATION_NUMBER").size().reset_index(name = "count").mean()
```

Out[494]:
```
count    5555.555556
dtype: float64
```

In [13]:
```
hydro_df.shape
```

Out[13]:
```
(150001, 13)
```

Build out code using sample file hydro_daily_mean_39. Get peak (maximum) values for same dates

To introduce a new flood exposure index, annual maximum peak flows were extracted from data for the gauging stations. Use this timeseries data to calculate the magnitude of the 100-year flood by using frequency analysis.

Analyze 100 years of flood data

Load 100 year sflood data

In [39]:
```
%%time
hydro_flood_100yr = pd.read_csv("../hydrometric-annual-peaks_100yr_ON.csv")
```

```
Wall time: 172 ms
```

In [63]:
```
hydro_flood_100yr.columns
```

Out[63]:
```
Index(['x', 'y', 'DATE', 'IDENTIFIER', 'STATION_NAME', 'STATION_NUMBER',
       'PROV_TERR_STATE_LOC', 'TIMEZONE_OFFSET', 'DATA_TYPE_EN',
       'DATA_TYPE_FR', 'PEAK_CODE_EN', 'PEAK_CODE_FR', 'UNITS_EN', 'UNITS_FR',
       'SYMBOL_EN', 'SYMBOL_FR', 'PEAK'],
      dtype='object')
```

Add new date related columns

In [41]:
```
%%time
hydro_flood_100yr["datetime"] = pd.to_datetime(hydro_flood_100yr["DATE"])
hydro_flood_100yr["date"] = hydro_flood_100yr["datetime"].dt.date
hydro_flood_100yr["year"] = hydro_flood_100yr["datetime"].dt.year
```

```
Wall time: 48.4 ms
```

Cleanup IDENTIFIER column for consistency

In [42]:
```
hydro_flood_100yr["IDENTIFIER"] = hydro_flood_100yr["IDENTIFIER"].str.split(".")
```

In [14]:
```
hydro_flood_100yr["IDENTIFIER"].head()
```

Out[14]:
```
0    [02AA002, 1973, flow-debit, maximum-maximale]
1    [02AA002, 1974, flow-debit, maximum-maximale]
2    [02AA002, 1975, flow-debit, maximum-maximale]
3    [02AA002, 1976, flow-debit, maximum-maximale]
4    [02AA002, 1978, flow-debit, maximum-maximale]
Name: IDENTIFIER, dtype: object
```

In [43]:
```
identifier = []
for id in range(len(hydro_flood_100yr["IDENTIFIER"])):
    identifier.append(hydro_flood_100yr["IDENTIFIER"][id][0])
hydro_flood_100yr["IDENTIFIER"] = pd.Series(identifier)
```

In [23]:
```
### Years in which the minimum and maximum flow occurs - per station
```

In [44]:
```
%%time
hydro_flood_100yr_minmax_yr = hydro_flood_100yr[(hydro_flood_100yr["PEAK_CODE_EN"] == "Maximum") & \
                                                 (hydro_flood_100yr["DATA_TYPE_EN"] == "Flow")].
groupby(["x", "y", "STATION_NAME"], \
                                                as_index = False)["year"].agg(["count", "min", \
                                                                                "max"]).sort_valu
es("count", \

ascending = False).apply(lambda x: \

(x).astype("int")).reset_index()
```

```
Wall time: 31.2 ms
```

In [490]:
```
hydro_flood_100yr_minmax_yr.head()
```

Out[490]:

	x	y	STATION_NAME	count	year_minflow	year_maxflow	100yr_mag_peakflow
0	-80.315750	43.353111	GRAND RIVER AT GALT	86	1930	2017	47.128018
1	-93.913361	48.634472	RAINY RIVER AT MANITOU RAPIDS	68	1930	2015	47.128018
2	-81.208580	42.973560	THAMES RIVER NEAR EALING	67	1938	2018	47.128018
3	-80.994858	43.059109	MIDDLE THAMES RIVER AT THAMESFORD	64	1948	2018	47.128018
4	-81.331779	42.962502	THAMES RIVER AT BYRON	64	1938	2018	47.128018

In [45]:
```
hydro_flood_100yr_minmax_yr.rename(columns = {"min": "year_minflow", "max": "year_maxflow"}, inplace
= True)
```

Frequency analysis of data via Fast Fourier Transform (FFT)

Use 100 years of data. Ideally the past 20 years should be used given the guidelines. 100 years gives more data and is therefore easier for proof of concept (POC).

In [46]:
```
hydro_flood_100yr.head()
```

Out[46]:

	x	y	DATE	IDENTIFIER	STATION_NAME	STATION_NUMBER	PROV_TERR_STATE_LOC	TIMEZONE_OFF
0	-89.533333	48.07222	1973-11-21T16:30	02AA002	PINE RIVER NEAR CROOKS	02AA002	ON	
1	-89.533333	48.07222	1974-05-11T22:18	02AA002	PINE RIVER NEAR CROOKS	02AA002	ON	
2	-89.533333	48.07222	1975-04-25T22:57	02AA002	PINE RIVER NEAR CROOKS	02AA002	ON	
3	-89.533333	48.07222	1976-04-16T12:00	02AA002	PINE RIVER NEAR CROOKS	02AA002	ON	
4	-89.533333	48.07222	1978-06-26T02:05	02AA002	PINE RIVER NEAR CROOKS	02AA002	ON	

Fast Fourier Transform (FFT) to extract magnitude of peakflow

In [47]:
```
def fft_station_peakflow2(peaks_df, nmax):

    get_df = peaks_df
    peaks = np.array(peaks_df["PEAK"])
    results = np.fft.fft(peaks, nmax, norm = "ortho")
    results = np.ndarray.max(np.abs(results))
    get_df["mag_100yr_peakflow"] = results

    return get_df
```

Magnitude of peakflow over 100 years

In [48]:
```
n = hydro_flood_100yr_minmax_yr["year_maxflow"] - hydro_flood_100yr_minmax_yr["year_minflow"] + 1
nmax = max(n)
peakflow_100yr_df = hydro_flood_100yr.groupby("STATION_NAME")["STATION_NAME", \
                                                        "PEAK", "year"].apply(lambda x: fft_stati
on_peakflow2(x, nmax))

hydro_flood_100yr_minmax_yr["100yr_mag_peakflow"] = peakflow_100yr_df["mag_100yr_peakflow"]
```

In [445]:
```
peakflow_100yr_df.columns
```

Out[445]:
```
Index(['STATION_NAME', 'PEAK', 'year', 'mag_100yr_peakflow'], dtype='object')
```

In [140]:
```
mag_100yr_peakflow2 = peakflow_100yr_df[["STATION_NAME", "mag_100yr_peakflow"]].groupby("STATION_NA
ME")["mag_100yr_peakflow"].agg("first").reset_index()
```

Remove any magnitude peakflow null values

In [141]:
```
mag_100yr_peakflow2 = mag_100yr_peakflow2 = mag_100yr_peakflow2[mag_100yr_peakflow2["mag_100yr_peak
flow"].isnull() == False]
hydro_flood_100yr = hydro_flood_100yr[hydro_flood_100yr["PEAK"].isnull() == False]
```

Add id column for ease of plotting

In [153]:
```
mag_100yr_peakflow2.sort_values("mag_100yr_peakflow", ascending = True, inplace = True)
mag_100yr_peakflow2["id_value"] = np.arange(len(mag_100yr_peakflow2)) + 1
```

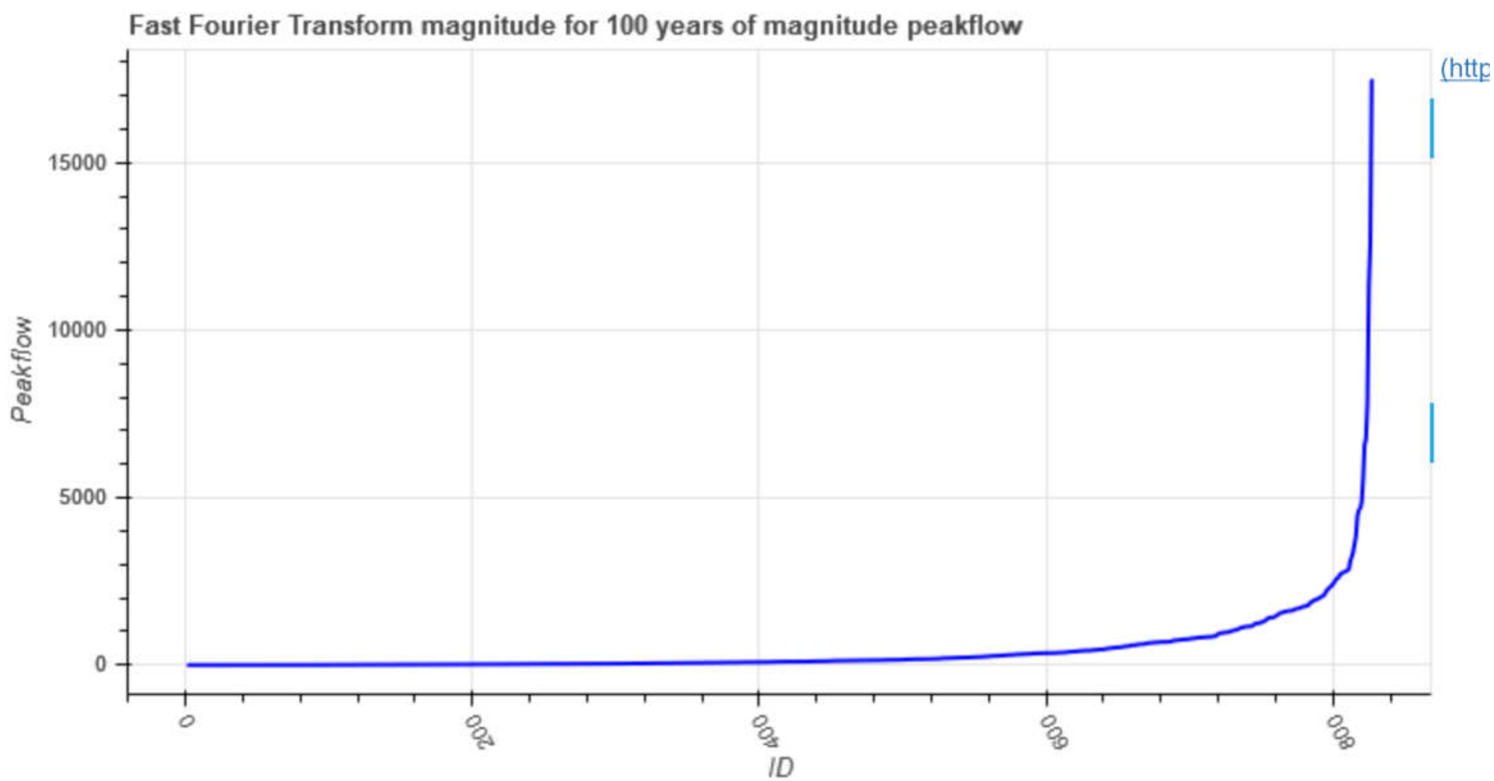
In [154]:
```
mag_100yr_peakflow2.head()
```

Out[154]:

	STATION_NAME	mag_100yr_peakflow	id_value
301	LAKE 120 OUTLET NEAR KENORA	0.006076	1
309	LAKE 230 OUTLET NEAR KENORA	0.006716	2
179	EASTERN TRIBUTARY TO DASHWA LAKE NEAR ATIKOKAN	0.007569	3
178	EAST TRIBUTARY TO PERCH LAKE INLET NO. 2 NEAR ...	0.012259	4
311	LAKE 239 NORTHEAST INLET NEAR KENORA	0.018229	5

In [190]:
```
#output_notebook()
output_notebook()

session_data = ColumnDataSource(data = mag_100yr_peakflow2)


id_value = session_data.data['id_value'].tolist()

plot = figure(plot_width=750, plot_height = 400, \
              min_border = 0)
plot.line(x = "id_value", y = "mag_100yr_peakflow", source = session_data, \
          line_width = 2, line_color = "blue")
plot.title.text = 'Fast Fourier Transform magnitude for 100 years of magnitude peakflow'
plot.xaxis.axis_label = 'ID'
plot.yaxis.axis_label = 'Peakflow'
plot.xaxis.major_label_orientation = 90

hover = HoverTool(tooltips = [('ID', '@id_value'), ('Magnitude 100 year Peak Flow', '@mag_100yr_pea
kflow')])
plot.add_tools(hover)

show(plot)
```

(https://bokeh.pydata.org) Loading BokehJS ...



In [169]:
```
(mag_100yr_peakflow2[mag_100yr_peakflow2["mag_100yr_peakflow"] > outlier].count()/mag_100yr_peakflo
w2["mag_100yr_peakflow"].count().astype(float)).id_value
```

Out[169]: 0.5332527206771464

53% of the peak flow magnitudes are outliers, hence the graph appearing to have lots of values close to 0. Too many values to remove from analysis (over half of the values). Solution:-

```
- Keep outliers
- Transofrm y axis by taking log10
```

In [ ]:

```

```

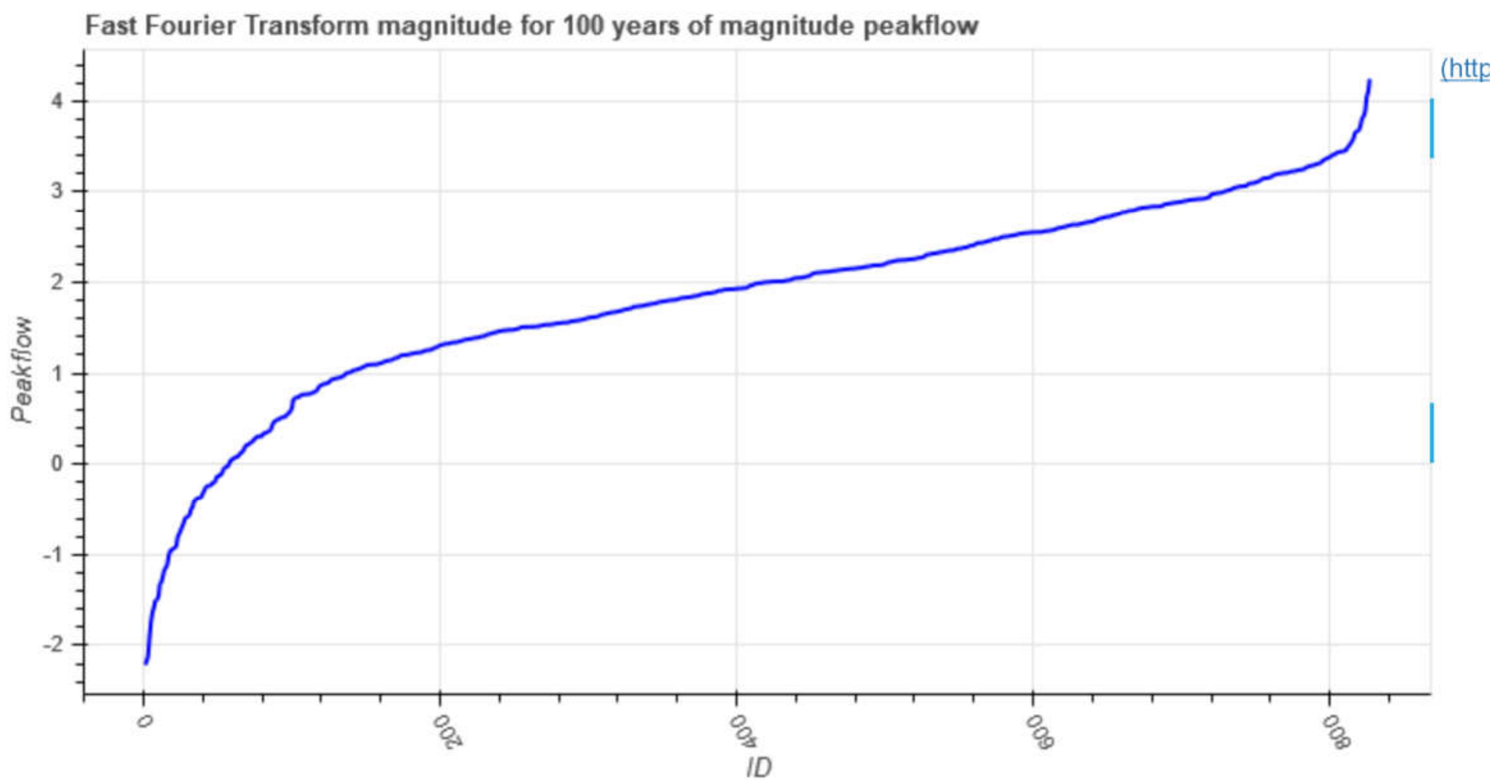
Transform the magnitude by log10

In [177]:

```
mag_100yr_peakflow2["log_mag_100yr_peakflow"] = np.log10(mag_100yr_peakflow2["mag_100yr_peakflow"])
```

In [189]:
```
output_notebook()

session_data = ColumnDataSource(data = mag_100yr_peakflow2)


id_value = session_data.data['id_value'].tolist()

plot = figure(plot_width=750, plot_height = 400, \
              min_border = 0)#, Tooltips = tooltips)
plot.line(x = "id_value", y = "log_mag_100yr_peakflow", source = session_data, \
          line_width = 2, line_color = "blue")
plot.title.text = 'Fast Fourier Transform magnitude for 100 years of magnitude peakflow'
plot.xaxis.axis_label = 'ID'
plot.yaxis.axis_label = 'Peakflow'
plot.xaxis.major_label_orientation = 90

hover = HoverTool(tooltips = [('ID', '@id_value'), ('Magnitude 100 year Peak Flow', '@mag_100yr_pea
kflow')])
plot.add_tools(hover)

show(plot)
```

(https://bokeh.pydata.org) Loading BokehJS ...




```
In [182]: mag_100yr_peakflow2.shape
```

```
Out[182]: (827, 4)
```

Create actual flood index

Load annual mean peakflows

In [ ]:
```
Daily mean peakflow would be better than annual mean peakflows, however this is not available
for historical hydrometric data.
```

In [403]:
```
%%time
hydro_annual_peakflow = pd.read_csv("../hydrometric_daily_peaks_ON_47yr.csv")
```

```
Wall time: 115 ms
```

In [404]:
```
hydro_annual_peakflow.columns
```

Out[404]:
```
Index(['x', 'y', 'IDENTIFIER', 'STATION_NAME', 'STATION_NUMBER',
       'PROV_TERR_STATE_LOC', 'DATA_TYPE_EN', 'DATA_TYPE_FR', 'MAX_DATE',
       'MAX_SYMBOL_EN', 'MAX_SYMBOL_FR', 'MAX_VALUE', 'MIN_DATE',
       'MIN_SYMBOL_EN', 'MIN_SYMBOL_FR', 'MIN_VALUE'],
      dtype='object')
```

In [61]:
```
### Select the most important columns
```

In [406]:
```
hydro_annual_peakflow_cols = hydro_annual_peakflow[['x', 'y', #'IDENTIFIER', \
                                                    'STATION_NAME', \
                                                    'STATION_NUMBER', \
       'PROV_TERR_STATE_LOC', 'DATA_TYPE_EN', 'MAX_DATE',
       'MAX_SYMBOL_EN', 'MAX_VALUE']]
```

Rename columns

In [407]:
```
hydro_annual_peakflow_cols = hydro_annual_peakflow_cols.rename(columns = {"MAX_DATE": "DATE_annual_
peakflow", \
                                                  "MAX_VALUE": "annual_avg_peakflow", \
                                      "DATA_TYPE_EN": "flow_or_level"})

# Extract datetime and year
hydro_annual_peakflow_cols["annual_peakflow_datetime"] = pd.to_datetime(hydro_annual_peakflow_cols
["DATE_annual_peakflow"])
hydro_annual_peakflow_cols["flow_year"] = hydro_annual_peakflow_cols["annual_peakflow_datetime"].d
t.year.astype(int)
```

Join magnitude 100yr peakflow data and annual peak flow. Remove last row (blank row)

In [448]:
```
%%time
hydro_index_data  = mag_100yr_peakflow2.merge(hydro_annual_peakflow_cols, \
                                              how = "inner", \
                                                on = "STATION_NAME")
```

```
Wall time: 25 ms
```

In [493]:
```
hydro_index_data.columns
```

Out[493]:
```
Index(['STATION_NAME', 'mag_100yr_peakflow', 'id_value',
       'log_mag_100yr_peakflow', 'x', 'y', 'STATION_NUMBER',
       'PROV_TERR_STATE_LOC', 'flow_or_level', 'DATE_annual_peakflow',
       'MAX_SYMBOL_EN', 'annual_avg_peakflow', 'annual_peakflow_datetime',
       'flow_year'],
      dtype='object')
```

In [455]:
```
flood_index = (hydro_index_data["mag_100yr_peakflow"]/(hydro_index_data["annual_avg_peakflow"].astype(float)))
```

In [456]:
```
flood_index.replace([np.inf, -np.inf], np.nan, inplace = True)
```

In [457]:
```
flood_index_log = np.log10(flood_index)
```

In [ ]:
```
#### Remove any null values from the index
```

In [464]:
```
num_nulls = np.count_nonzero(pd.Series.isnull(flood_index_log).values == True)
print("Number of null values =", num_nulls)
```

```
Number of null values = 10
```

In [465]:
```
flood_index_log = flood_index_log[~np.isnan(flood_index_log)].sort_values(ascending = True)
```

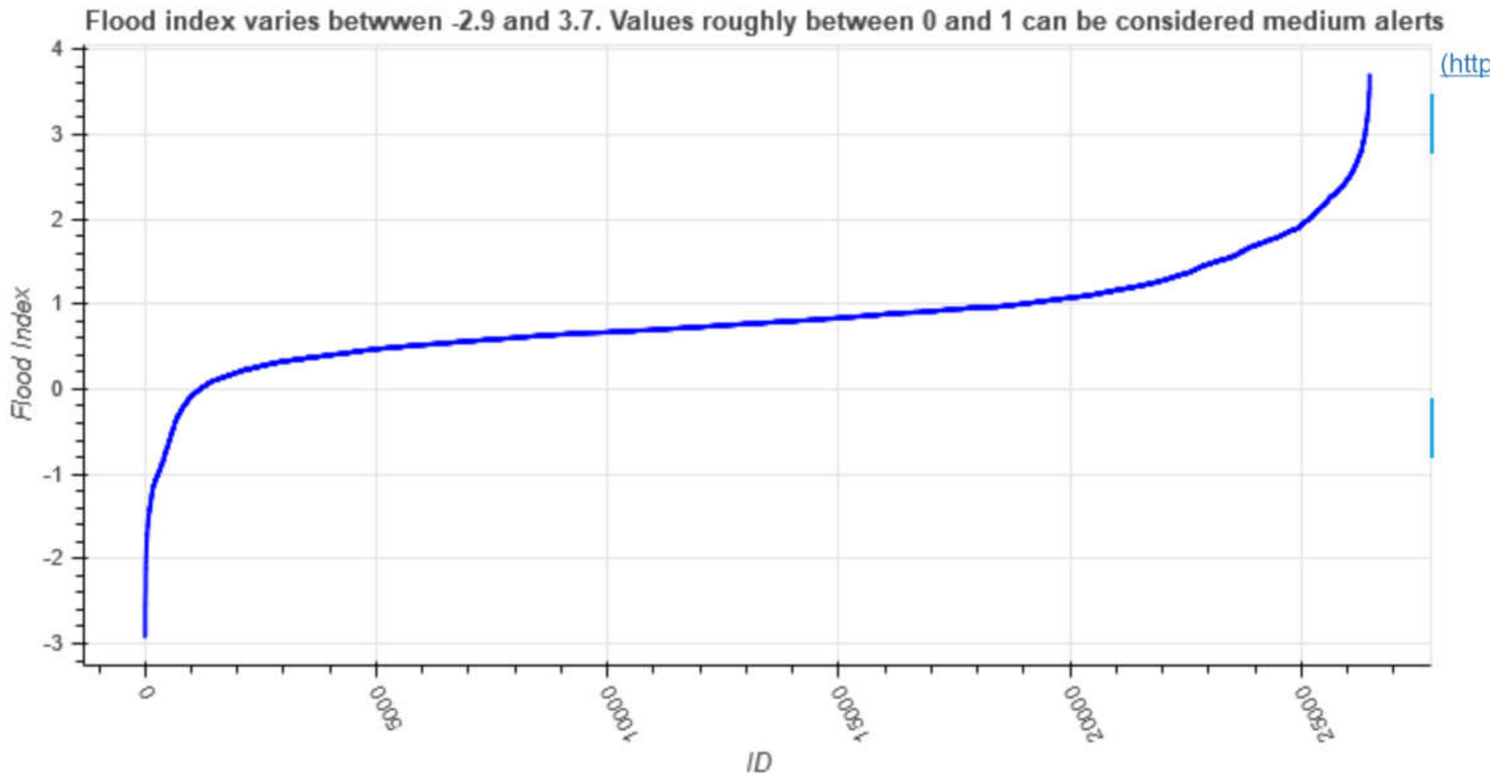
In [483]:
```
round(flood_index_log.quantile([0, 0.25, 0.5, 0.75, 1.0]), 3)
```

Out[483]:
```
0.00   -2.925
0.25    0.546
0.50    0.773
0.75    1.063
1.00    3.717
dtype: float64
```

In [468]:
```
flood_index_dict = {"flood_index": flood_index_log, "id_value": np.arange(len(flood_index_log)) +
1}
flood_index_log_df = pd.DataFrame(flood_index_dict)
```

In [489]:
```
#output_notebook()
output_notebook()

session_data = ColumnDataSource(data = flood_index_log_df)

id_value = session_data.data['id_value'].tolist()

plot = figure(plot_width=750, plot_height = 400, \
              min_border = 0)#, Tooltips = tooltips)
plot.line(x = "id_value", y = "flood_index", source = session_data, \
          line_width = 2, line_color = "blue")
plot.title.text = 'Flood index varies betwwen -2.9 and 3.7. Values roughly between 0 and 1 can be c
onsidered medium alerts'
plot.xaxis.axis_label = 'ID'
plot.yaxis.axis_label = 'Flood Index'
plot.xaxis.major_label_orientation = 90

hover = HoverTool(tooltips = [('ID', '@id_value'), ('Flood Index', '@flood_index')])
plot.add_tools(hover)

show(plot)
```

(https://bokeh.pydata.org) BokehJS 1.3.4 successfully loaded.




```
In [ ]: A good approximation for index can be any values between 0 and 1 are medium alerts.
        Less than 0 for low alerts, and > 1 high
```

In [520]:

```
#output_file("alert_table.html")
alert_data = dict( \
            alert = ["Low", "Medium", "High"], \
            index = ["-2.9 to 0", "0 to 1.06", "1.06 to 3.72"], \
            approx_index = ["< 0", "0 to 1", "> 1"]
           )

source_data = ColumnDataSource(alert_data)

alert_columns = [TableColumn(field = "alert", title = "Alert Category", width = 100), \
                TableColumn(field = "index", title = "Actual Flood Index", width = 100), \
                TableColumn(field = "approx_index", title = "Modified Flood Index", width = 100)]

alert_table = DataTable(source = source_data, \
                       columns = alert_columns, \
                       fit_columns = True, selectable = True, sortable = True, \
                        width = 400, height = 400)

show(widgetbox(alert_table, height = 100))
```

```
WARNING:bokeh.core.validation.check:W-1005 (FIXED_SIZING_MODE): 'fixed' sizing mode requires width
and height to be set: WidgetBox(id='10215', ...)
```

#	Alert Category	Actual Flood Index	Modified Flood Index
0	Low	-2.9 to 0	< 0
1	Medium	0 to 1.06	0 to 1
2	High	1.06 to 3.72	> 1

```
WARNING:bokeh.core.validation.check:W-1005 (FIXED_SIZING_MODE): 'fixed' sizing mode requires width
and height to be set: WidgetBox(id='10215', ...)
```

In [ ]:

```
Modified flood index should generalise well to othe hydrometric data.
```

Future work

In [ ]:
```
Modify the following, creating the index from 1 and 2 :
1. Use 20 year historical time series data instead of 100 year
2. Daily mean peakflow or daily peakflow - instead of annual mean peakflow
3. Get 48 hour forecasts based on equations - See MeteoHAck2019.pptx and
    MeteoHack2019 - create index.pptx.
    - Daily mean peakflow and/or 5 minute real time data can be used to build this model.
    - Techniques to experiment with ARIMA, RNN etc.
4. Create a seperate model for predicting flood alerts. Note both models will need to take
    water levels into consideration something that was not done here.
```

In [ ]:
```
###########################################################################################
```