# Data Analysis

## 1 Mounting Google Drive

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

## 2 Imports

```
[ ]: import pandas as pd
     import numpy as np
     from sklearn.feature_selection import SelectKBest, chi2
     from sklearn.preprocessing import MinMaxScaler
     from sklearn.tree import DecisionTreeClassifier
     import matplotlib.pyplot as plt
     import seaborn as sns
     import os
     from pprint import pprint
```

## 3 Reading Data

```
[ ]: DATA_DIR = '/content/drive/My Drive/Colab Notebooks/Intrusion Detection/data/
       ↪NSL-KDD'
```

```
[ ]: DATA_COLS = [
         'Duration',
         'Src Bytes',
         'Dst Bytes',
         'Land',
         'Wrong Fragment',
         'Urgent',
         'Hot',
         'Num Failed Logins',
    'Logged In',
    'Num Compromised',
    'Root Shell',
    'Su Attempted',
    'Num Root',
    'Num File Creations',
    'Num Shells',
    'Num Access Files',
    'Num Outbound Cmds',
    'Is Hot Logins',
    'Is Guest Login',
    'Count',
    'Srv Count',
    'Serror Rate',
    'Srv Serror Rate',
    'Rerror Rate',
    'Srv Rerror Rate',
    'Same Srv Rate',
    'Diff Srv Rate',
    'Srv Diff Host Rate',
    'Dst Host Count',
    'Dst Host Srv Count',
    'Dst Host Same Srv Rate',
    'Dst Host Diff Srv Rate',
    'Dst Host Same Src Port Rate',
    'Dst Host Srv Diff Host Rate',
    'Dst Host Serror Rate',
    'Dst Host Srv Serror Rate',
    'Dst Host Rerror Rate',
    'Dst Host Srv Rerror Rate',
    'Protocol Type',
    'Service',
    'Flag'
]
```

```
[ ]: train_data = pd.read_csv(os.path.join(DATA_DIR, 'KDDTrain+.txt'), header=None)
```

```
[ ]: train_data.head()
```

## 4 Data Segregation

```
[ ]: numeric_features = pd.concat((train_data.iloc[:,0], train_data.iloc[:,4:41]),␣
      ↪axis=1)
     nominal_features = train_data.iloc[:,1:4]
     labels = train_data.iloc[:,41]
```

```
[ ]: numeric_cols = DATA_COLS[:-3]
     nominal_cols = DATA_COLS[-3:]
```

```
[ ]: numeric_features.head()
```

```
[ ]: nominal_features.head()
```

```
[ ]: labels.head()
```

```
[ ]: numeric_features_np = numeric_features.to_numpy()
     nominal_features_np = nominal_features.to_numpy()
     labels_np = labels.to_numpy()
```

### 4.1 MinMaxScaler

```
[ ]: min_max_scaler = MinMaxScaler()
```

```
[ ]: numeric_features_scaled = min_max_scaler.fit_transform(numeric_features_np)
```

### 4.2 Get Dummies

```
[ ]: nominal_features_one_hot = pd.get_dummies(nominal_features)
```

```
[ ]: nominal_features_one_hot.head()
```

```
[ ]: nominal_features_one_hot_np = nominal_features_one_hot.to_numpy()
```

## 5 Numeric Features

### 5.1 Univariate Selection

```
[ ]: select_k_best = SelectKBest(score_func=chi2, k='all')
```

```
[ ]: k_best = select_k_best.fit(numeric_features_scaled, labels_np)
```

```
[ ]: numeric_feature_univariate_selection = pd.Series(k_best.scores_,␣
      ↪index=numeric_cols)
```

[ ]:
```python
top_n = 10
title = f'Univariate Selection - Top {top_n} Features'

fig = plt.figure()
numeric_feature_univariate_selection.nlargest(top_n).plot(
    kind='bar',
    figsize=(5,5),
    color='k'
    # title=title
)

plt.ylabel('$Chi^2$ Scores')
plt.xlabel('Attributes')

fig.tight_layout()

plt.savefig(title + '.png')
```

[ ]:
```python
last_n = 10
title = f'Univariate Selection - Last {last_n} Features'

fig = plt.figure()
numeric_feature_univariate_selection.nsmallest(last_n).plot(
    kind='bar',
    figsize=(5,5),
    color='k'
    # title=title
)

plt.ylabel('$Chi^2$ Scores')
plt.xlabel('Attributes')

fig.tight_layout()

plt.savefig(title + '.png')
```

## 5.2 Correlation Matrix with Heatmap

[ ]:
```python
def plot_heatmap(features, cols, thresh=0.75, figsize=(20,20)):
    features_pd = pd.DataFrame(features)
    correlation_mat = features_pd.corr()

    top_corr_features = set()
    for i in range(38):
        for j in range(i+1,38):
            if abs(correlation_mat[i][j]) >= thresh:
                top_corr_features.add(i)
                top_corr_features.add(j)

    top_corr_features_list = list(top_corr_features)
    top_correlation_mat = features_pd.iloc[:, top_corr_features_list].corr()

    top_corr_cols = [cols[i] for i in top_corr_features_list]

    short_col_list = list()
    short_cols = []
    ch = 'A'
    for i in range(len(top_corr_cols)):
        short_col_list.append([chr(ord(ch) + i), top_corr_cols[i]])
        short_cols.append(chr(ord(ch) + i))

    fig = plt.figure(figsize=figsize)
    sns.heatmap(top_correlation_mat, xticklabels=short_cols,␣
 ↪yticklabels=short_cols, annot=True, cmap=plt.cm.Greys)

    title = f'Correlation Heatmap with {thresh*100}% Threshold'
    fig.tight_layout()

    plt.savefig(title + '.png')

    return short_col_list
```

[ ]:
```
short_col_list = plot_heatmap(numeric_features_scaled, numeric_cols, thresh=0.
 ↪95, figsize=(7,7))
```

[ ]:
```
short_col_list
```

# 6 Nominal Features

## 6.1 Decision Tree Classifier

[ ]:
```
nominal_features.head()
```

[ ]:
```
dtc = DecisionTreeClassifier()
dtc.fit(nominal_features_one_hot_np, labels_np)
```

[ ]:
```
weights = []

for col in nominal_features.columns:
    weights.append(nominal_features[col].value_counts().to_dict())
```

```python
for curr_val in weights:
    for key in curr_val.keys():
        curr_val[key] /= nominal_features.shape[0]

pprint(weights)
```

[ ]:
```python
importances = [0] * 3
for i in range(84):
    curr_column = nominal_features_one_hot.columns[i]
    importances[int(curr_column[0])-1] +=␣
 ↪weights[int(curr_column[0])-1][curr_column[2:]] * dtc.feature_importances_[i]
```

[ ]:
```python
importances
```

[ ]:
```python
importances_series = pd.Series(importances, index=DATA_COLS[-3:])

fig = plt.figure()
title = 'Feature Importances using Decision Tree Classifier'
importances_series.plot(
    kind='bar',
    # title=title,
    figsize=(5,5),
    color='k'
)

plt.ylabel('Weighted Gini Index Scores')
plt.xlabel('Attributes')

fig.tight_layout()

plt.savefig(title + '.png')
```

[ ]:
```python

```