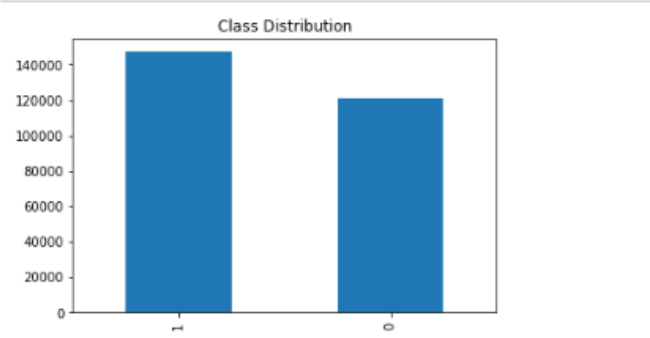
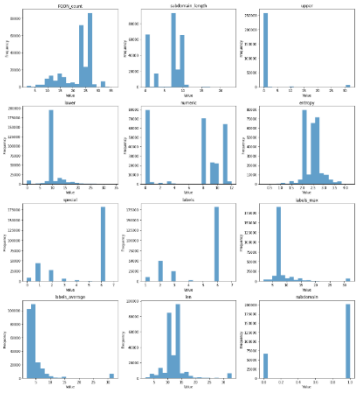


1. data imbalanced

as we can see that the class 1 that we interested in is look more than the class 0 from that we can say that our data is balanced

2. Statistical analysis of data

I made a visualization of the distribution of each feature's values using histograms, making it easier to understand the data distribution and identify patterns or outliers across multiple features simultaneously and use other statistical function like skew() and describe()



```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 268074 entries, 0 to 268073
Data columns (total 16 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   timestamp         268074 non-null  object
 1   FQDN_count        268074 non-null  int64
 2   subdomain_length  268074 non-null  int64
 3   upper             268074 non-null  int64
 4   lower             268074 non-null  int64
 5   numeric           268074 non-null  int64
 6   entropy           268074 non-null  float64
 7   special           268074 non-null  int64
 8   labels            268074 non-null  int64
 9   labels_max        268074 non-null  int64
 10  labels_average    268074 non-null  float64
 11  longest_word      268066 non-null  object
 12  sld               268074 non-null  object
 13  len               268074 non-null  int64
 14  subdomain         268074 non-null  int64
 15  Target Attack     268074 non-null  int64
dtypes: float64(2), int64(11), object(3)
memory usage: 32.7+ MB
```

```
# Check for data skewness
data.skew()
```

```
FQDN_count          -0.800315
subdomain_length    -0.313356
upper                6.633687
lower                0.562564
numeric             -0.304702
entropy             -0.103133
special             -0.566043
labels              -0.572677
labels_max           3.679053
labels_average       5.206518
longest_word         2.502320
sld                185.221263
len                  2.296762
subdomain           -0.883460
Target Attack        0.263657
dtype: float64
```

```
data.describe()
```

	FQDN_count	subdomain_length	upper	lower	numeric	entropy	special	labels	labels_max	labels_av
count	268074.000000	268074.000000	268074.000000	268074.000000	268074.000000	268074.000000	268074.000000	268074.000000	268074.000000	268074.0
mean	22.286596	6.059021	0.845420	10.410014	6.497586	2.485735	4.533577	4.788823	8.252233	4.8
std	6.001205	3.899505	4.941929	3.207725	4.499866	0.407709	2.187683	1.803256	4.415355	4.5
min	2.000000	0.000000	0.000000	0.000000	0.000000	0.219195	0.000000	1.000000	2.000000	2.0
25%	18.000000	3.000000	0.000000	10.000000	0.000000	2.054029	2.000000	3.000000	7.000000	3.1
50%	24.000000	7.000000	0.000000	10.000000	8.000000	2.570417	6.000000	6.000000	7.000000	3.6
75%	27.000000	10.000000	0.000000	10.000000	10.000000	2.767195	6.000000	6.000000	7.000000	4.0
max	36.000000	23.000000	32.000000	34.000000	12.000000	4.216847	7.000000	7.000000	32.000000	32.0

3.Data Cleansing and Feature creation

1- **First I checked for any null values** in the data and I found that their numbers in too low regarding to the length of our data so that I decided to drop it and also I dropped the timestamp column

2- **Second step is checking for any duplicate** in our data and I found approximately
I notices 91 thousands duplicate in data and this will lead low performance with the model so, I decided to drop them

3-third step after looking at all columns and each column datatype from info() function so I wanted to print each unique values in each column after that I found out that there is 2 columns that have string values with integer values in the two columns and I decided to treat with this problem by using function that useful for handling non-numeric values in specific columns by replacing them with the mode of those columns ,it might not be the best solution but this is the approach I thought

4.Feature Filtering with more than 2 methods

I used 3 methods (Anova,mutual_information,chi2) with 2 models logestic regression and random forest for the logestic regression model

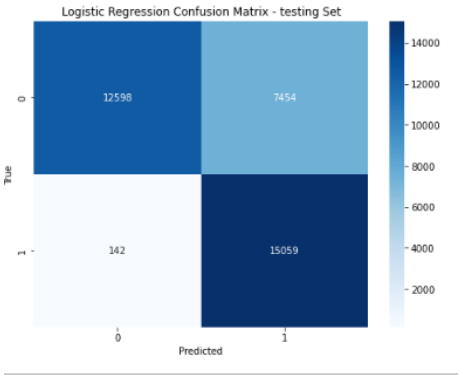
This is logestic baseline

logistic regression model

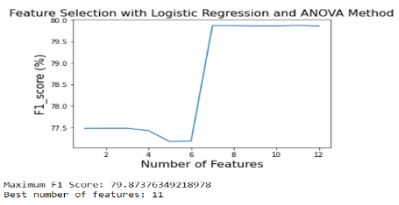
```
from sklearn.metrics import accuracy_score
from sklearn.metrics import accuracy_score, f1_score

lr_model = LogisticRegression(max_iter=1000)
lr_model.fit(X_train_scaled, y_train)
y_pred_test = lr_model.predict(X_test_scaled)
test_accuracy = accuracy_score(y_test , y_pred_test)
test_accuracy
print("accuracy",test_accuracy)
# Calculate F1 score
f1_lr = f1_score(y_test, y_pred_test)
print("F1 Score:", f1_lr)
```

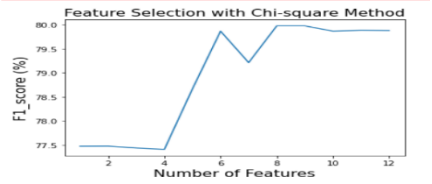
```
accuracy 0.7845289762573397
F1 Score: 0.7985893832529034
```



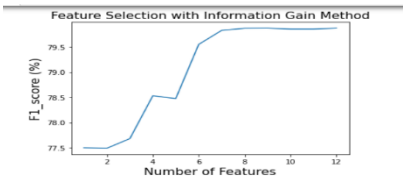
Then I applied logestic regression on the normalized data with the three methods to get the best accuracy with the lowest number of features




```
Maximum F1 Score: 79.87376349218978
Best number of features: 11
```

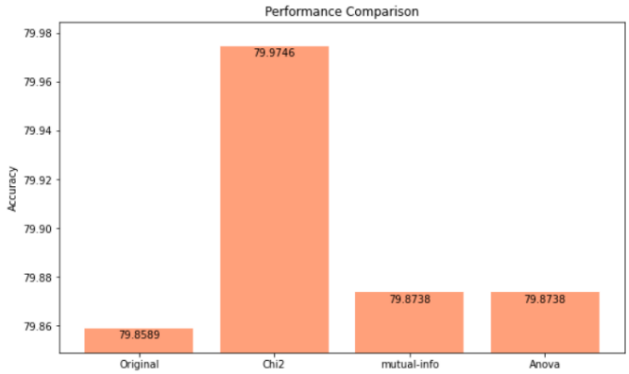



```
Maximum F1 Score: 79.97460451827946
Best number of features: 8
```




```
Maximum F1 Score: 79.87376349218978
Best number of features: 9
```

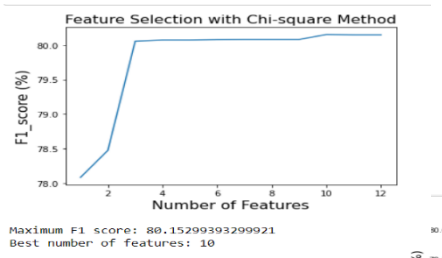
After that I applied a performance accuracy metric comparison to get the best accuracy with the best features that selected by the method form this comparison I found that logestic regression give the highest accuracy with chi2 with best number of features :8

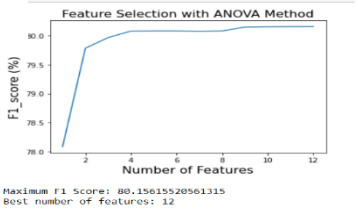


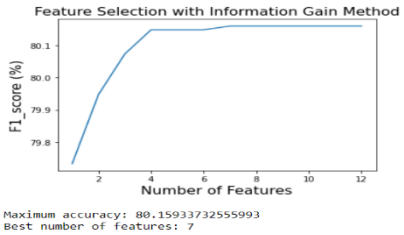
By the same way I applied random forest with the same three methods

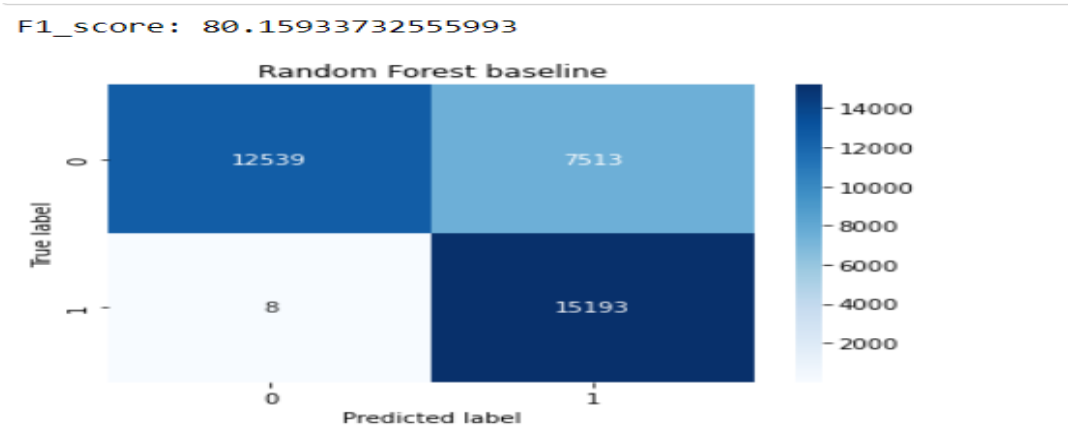
And get the highest accuracy with chi2 with f1 score **80.1530** % with best number of features of 10

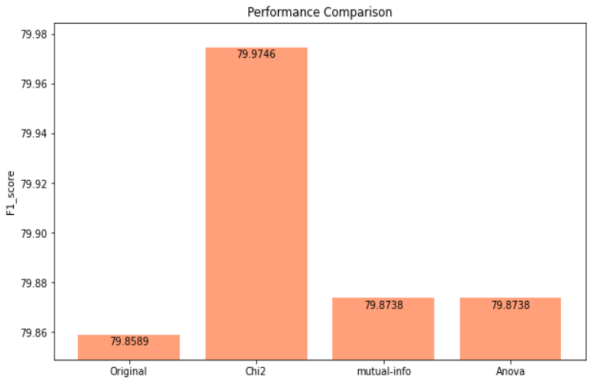
The Random Forest Baseline with f1 score 80.159%











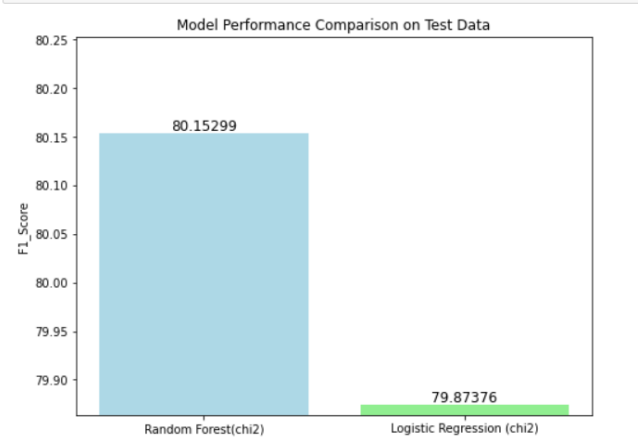
from the comparison chi2 is the highest f1 score with random forest model

5-Data Splitting and justification

I splitted the data using train test split function into train and test split sets with percent 80 % for training and 20% for testing

6-Choose and justify the correct performance metric

I chooses f1 score to be more accurate as f1 indicates how well our model can classify the target successfully and ploted the confusion matrix for the both 2 baseline models



champion model :

a quick comparison of model performance (F1 scores) on the test data, making it easy to see which model performed better as we can see in the graph the randomforest with 10 features gives us the best accuracy so that this is our champion model

after i choosed the random forest to be our champion model I made hyperparameter tunning using grid search to get the best parameters with the best values so I got that

```
Best Parameters: {'classifier__criterion': 'gini', 'classifier__max_depth'
: None, 'classifier__n_estimators': 100}
With Best F1 Score: 0.8043382562861086
```

After that I Created the pipeline with feature selection followed by the Random Forest classifier

The aim of this pipeline is to create a cohesive workflow:

Feature Selection: Use the chi-squared test to identify the top 10 most relevant features.

Classification: Train a Random Forest classifier on these selected features.

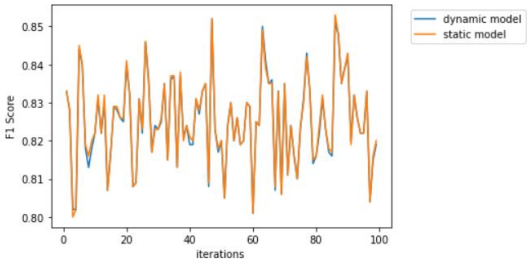
By chaining these steps in a pipeline, we efficiently perform feature selection and train a classifier, making it easier to reuse this workflow on new data. essentially saves the trained model (pipeline) as a binary file using pickle serialization. The saved file, named 'mutual_info_rf_model.pkl', will contain the serialized version of the trained model, allowing it to be loaded and used later without retraining.

Training reevaluation: First we make a list to take 1000 row of streaming data then we convert this list to dataframe after that using preprocessing the streaming data last using retrain function

Analyze the results obtained for both models

```
Window 0
Window 1
The F1 Score of Dynamic Model without retrain = 83.3%
The F1 of Static Model = 83.3%
**********
Window 2
The F1 Score of Dynamic Model without retrain = 82.8%
The F1 of Static Model = 82.8%
**********
Window 3
The F1 Score of Dynamic Model without retrain = 80.0%
 trained model on the new data
The f1 of Dynamic Model after retrain = 80.2%
The F1 of Static Model = 80.0%
```

Here I made model that compare both **Static** and **Dynamic** model for every streaming data 1000 by 1000 rows then we plot both Accuracies for first window they are both same then when accuracy lower from **threshold** that is = 81% a retrain model activates for Dynamic model it takes the stored past streaming data and train with it to enhance Dynamic model as we see



Then we plot the 200 iteration of streaming data to see performance of Each model as we see the **Dynamic** in lots of cases higher or above **Static** this indectes that Dynamic model keep enhanceing along time and thus its better to use Dynamic

Advantages/limitations and knowledge learned is clear and correct

Advantages of Static Models:

1. **Stability**: offer consistent predictions as they don't change over time.
2. **Simplicity:** easier to deploy and manage

Limitations of Static Models:

1. **Inability to Adapt:** cannot adapt to changes, less effective in highly streaming data
2. **Potential Degradation:** Over time, static models might experience degradation in performance cannot handle evolution.

Advantages of Dynamic Models:

1. **Adaptability:** Dynamic models can adapt and learn from new incoming data.
2. **Continuous Improvement:** can continually enhance their performance and accuracy

Limitations of Dynamic Models:

1. **Computational Cost** new data can be computationally expensive, with large volumes of streaming data.
2. **Potential Overfitting**: Continuous retraining might lead to overfitting