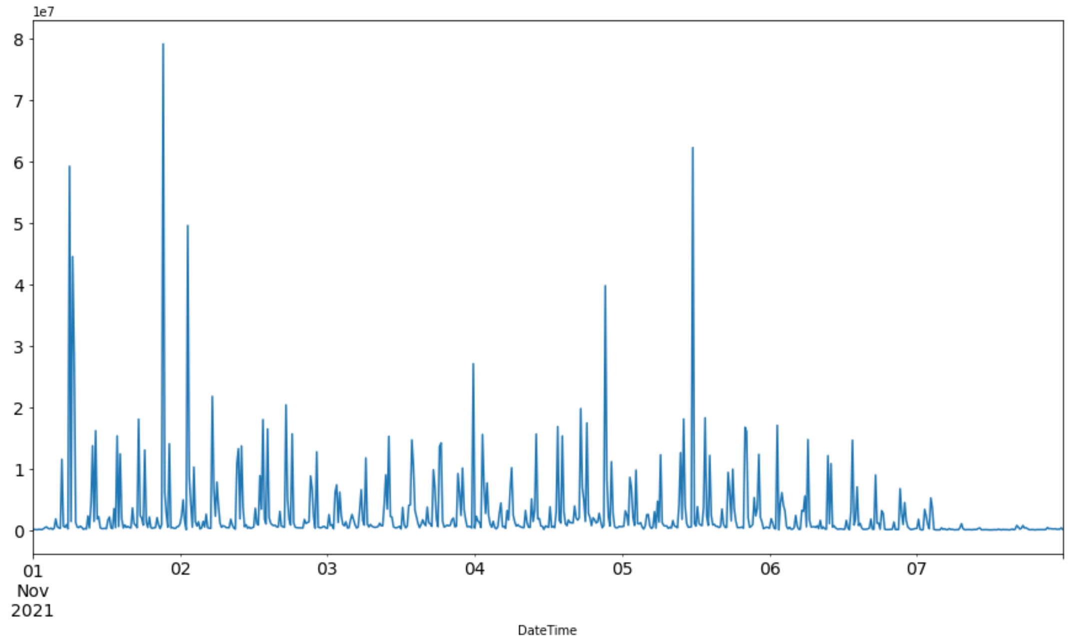
EDG Databricks Cluster Data Load Prediction Model Challenge

Challenges

When I started exploring the data the major problem was to create multiple models for different tables as I don't think it would be time efficient, thus I choose to use the 15 min interval total record count from all the tables as my input.

Some EDA and the final data –



Group data by 15 mins

In [11]:
```python
df = df.groupby(pd.Grouper(freq='15Min')).aggregate(np.sum)
```

In [12]:
```python
df
```

Out[12]:

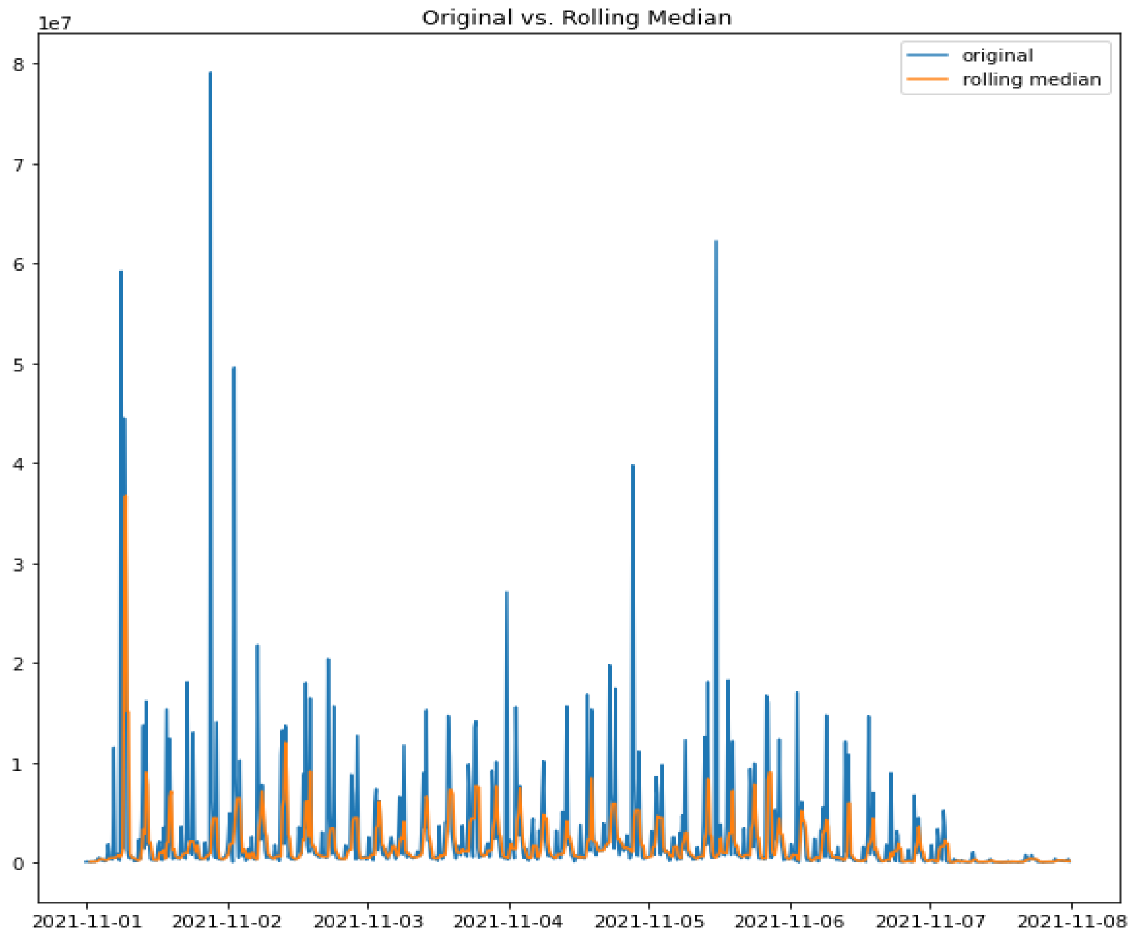
DateTime	TableRecordCount
2021-11-01 00:00:00	56929
2021-11-01 00:15:00	85724
2021-11-01 00:30:00	68082
2021-11-01 00:45:00	45065
2021-11-01 01:00:00	115849
...	...
2021-11-07 22:45:00	112995
2021-11-07 23:00:00	144090
2021-11-07 23:15:00	124498
2021-11-07 23:30:00	342549
2021-11-07 23:45:00	83540

672 rows × 1 columns

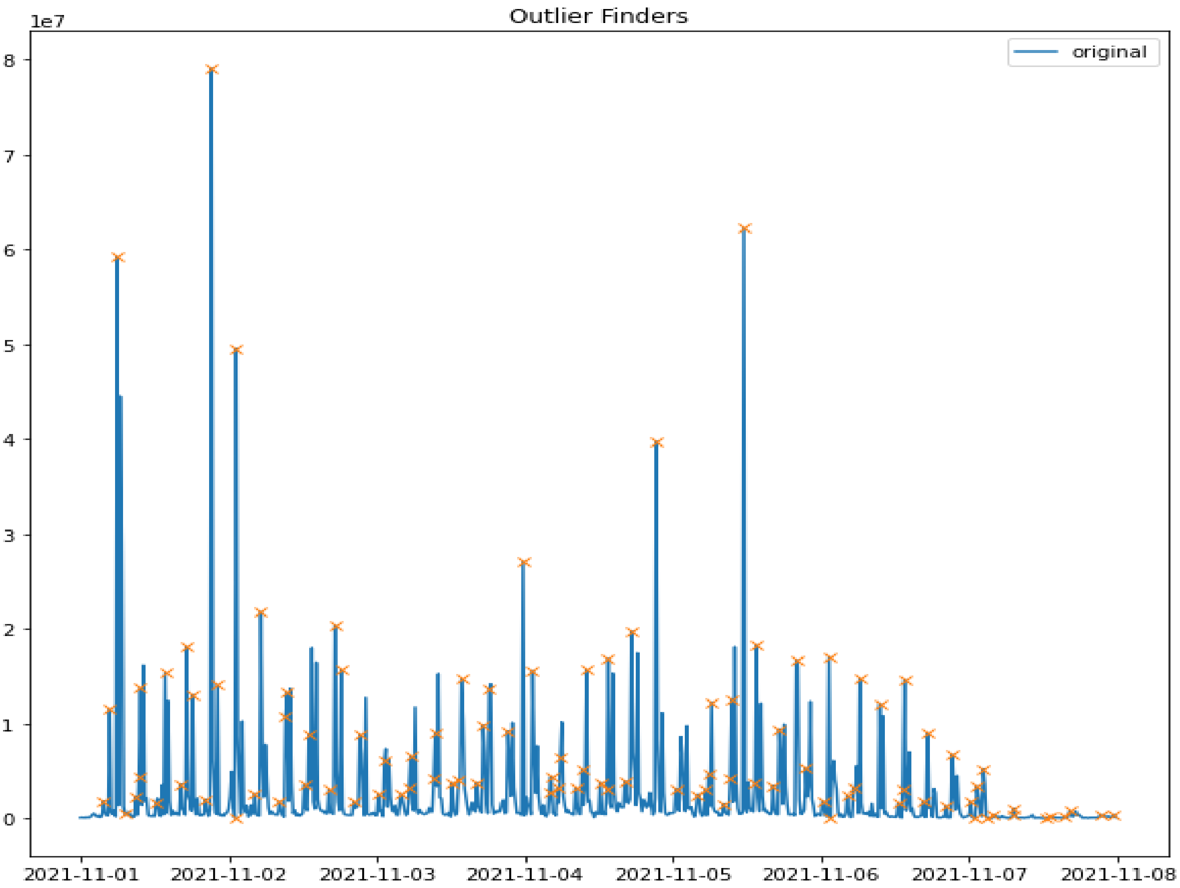
Solution Intuition

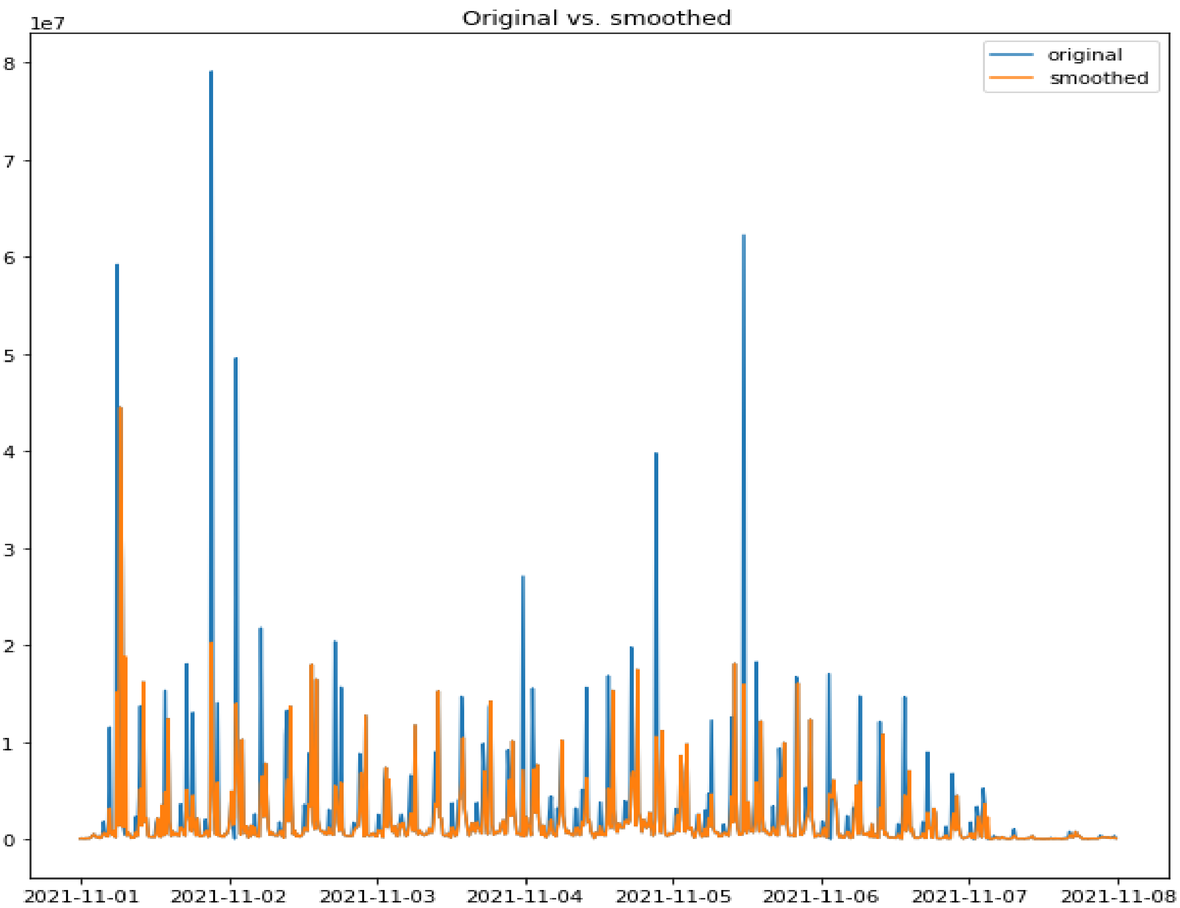
I started with removing outliers and smoothening the values for the model to better capture all the information.

Outlier identification by IForest and smoothing by rolling window median value



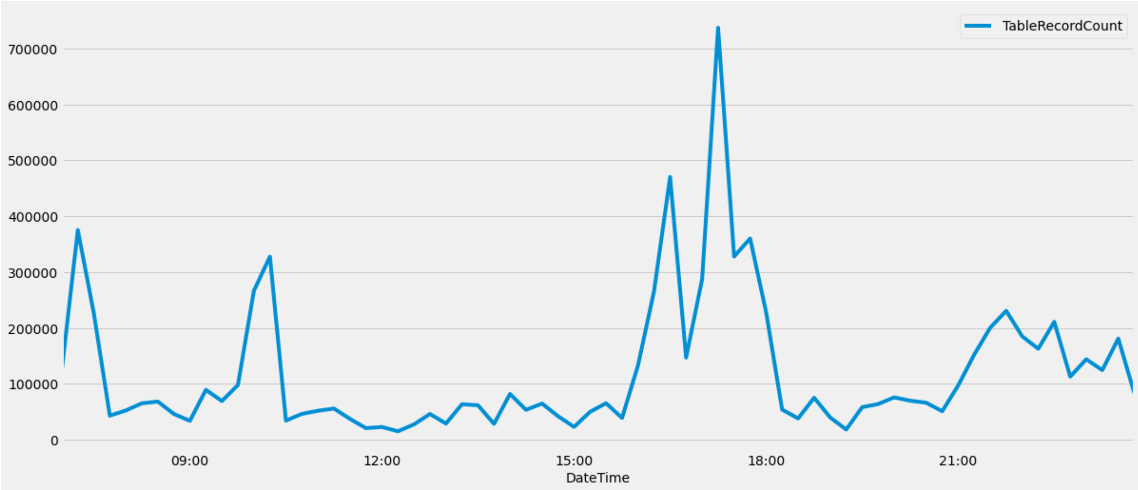

```
Outliers found at  DatetimeIndex(['2021-11-01 03:45:00', '2021-11-01 04:45:
00',
               '2021-11-01 06:00:00', '2021-11-01 07:15:00',
               '2021-11-01 09:00:00', '2021-11-01 09:30:00',
               '2021-11-01 09:45:00', '2021-11-01 12:15:00',
               '2021-11-01 13:45:00', '2021-11-01 16:15:00',
               ...
               '2021-11-07 03:00:00', '2021-11-07 04:00:00',
               '2021-11-07 07:00:00', '2021-11-07 07:15:00',
               '2021-11-07 12:30:00', '2021-11-07 13:15:00',
               '2021-11-07 15:30:00', '2021-11-07 16:15:00',
               '2021-11-07 21:15:00', '2021-11-07 23:30:00'],
            dtype='datetime64[ns]', name='DateTime', length=101, freq=None)
```

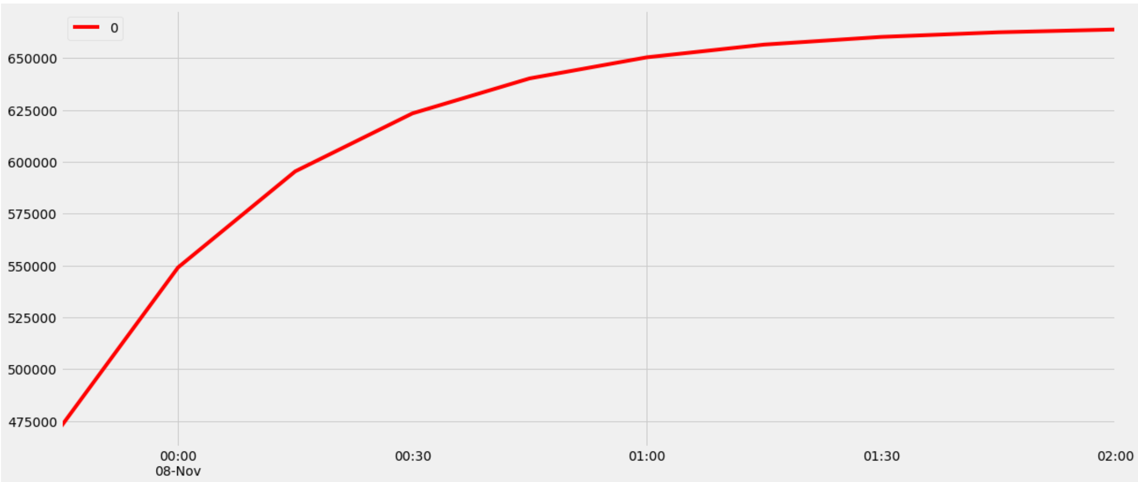

Outlier Finders
1e7
original
8
7
6
5
4
3
2
1
0
2021-11-01
2021-11-02
2021-11-03
2021-11-04
2021-11-05
2021-11-06
2021-11-07
2021-11-08


Original vs. smoothed
1e7
original
smoothed
8
7
6
5
4
3
2
1
0
2021-11-01
2021-11-02
2021-11-03
2021-11-04
2021-11-05
2021-11-06
2021-11-07
2021-11-08

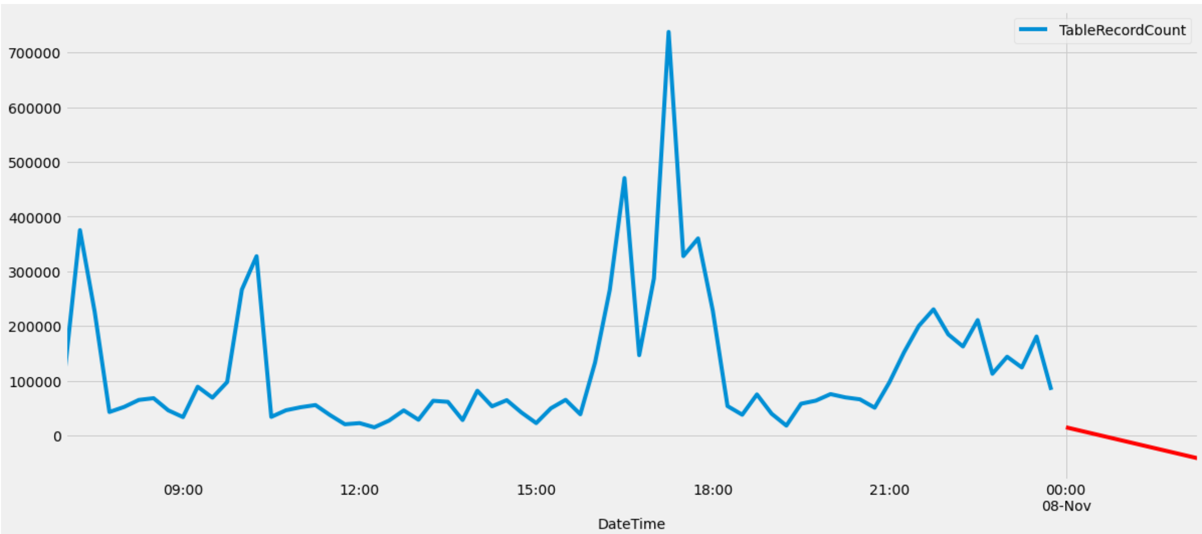
Model Experimentation

I check pdarima model then NeuralProphet AUTOML which were giving poor RMSE.

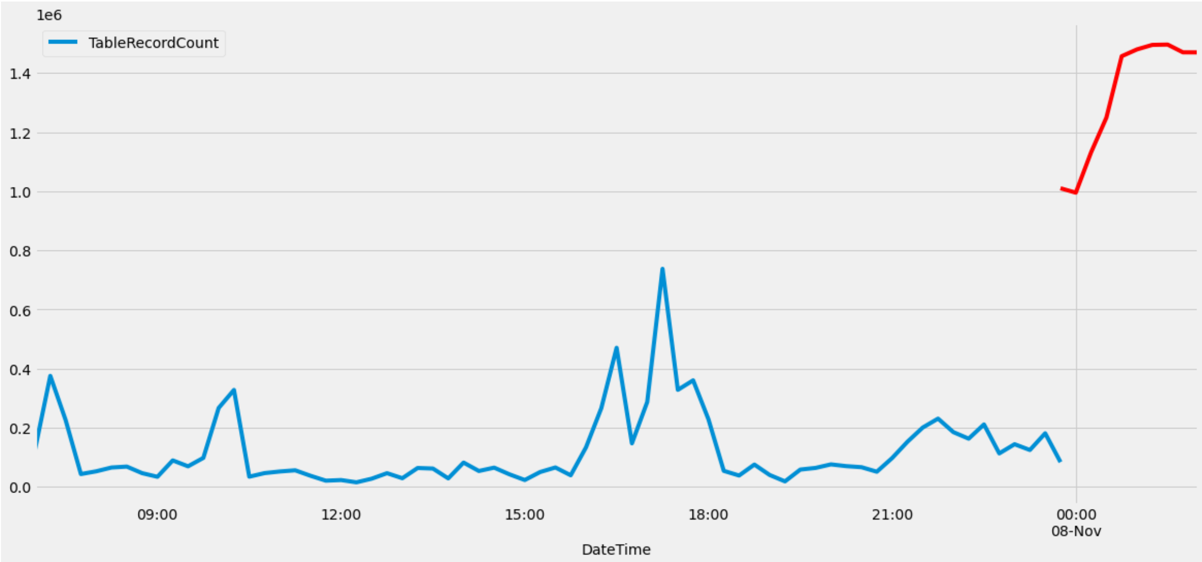




I also played with holtwinters with ExponentialSmoothing



I also played with AutoRegression Models



Using various Regression Models –

```
In [46]: from sklearn.linear_model import LinearRegression
from sklearn.neural_network import MLPRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
from sklearn.model_selection import TimeSeriesSplit,cross_val_score


# Spot Check Algorithms
models = []
models.append(('LR', LinearRegression()))
models.append(('NN', MLPRegressor(solver = 'lbfgs')))  #neural network
models.append(('KNN', KNeighborsRegressor()))
models.append(('RF', RandomForestRegressor(n_estimators = 10))) # Ensemble method - collection of many decision trees
models.append(('SVR', SVR(gamma='auto'))) # kernel = linear
# Evaluate each model in turn
results = []
names = []
for name, model in models:
    # TimeSeries Cross validation
    tscv = TimeSeriesSplit(n_splits=10)

    cv_results = cross_val_score(model, X_train, y_train, cv=tscv, scoring=rmse_score)

    results.append(cv_results)
    names.append(name)
    print('%s: %f (%f)' % (name, cv_results.mean(), cv_results.std()))

# Compare Algorithms
plt.boxplot(results, labels=names)
plt.title('Algorithm Comparison')
plt.show()
```

RMSE SCORES

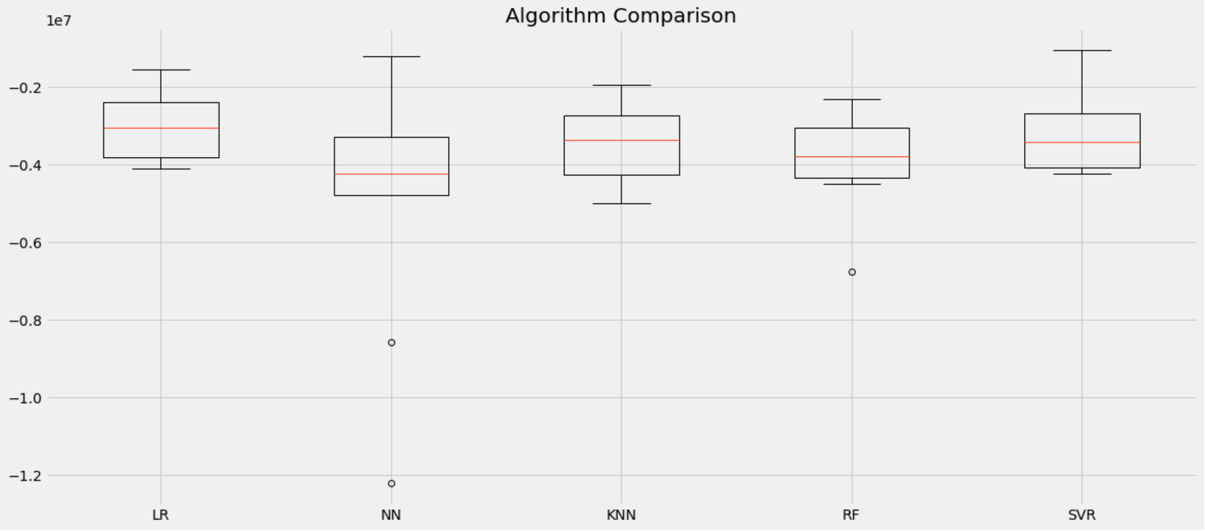
LR: -3036694.647225 (819607.162414)

NN: -4916930.785882 (3060292.698712)

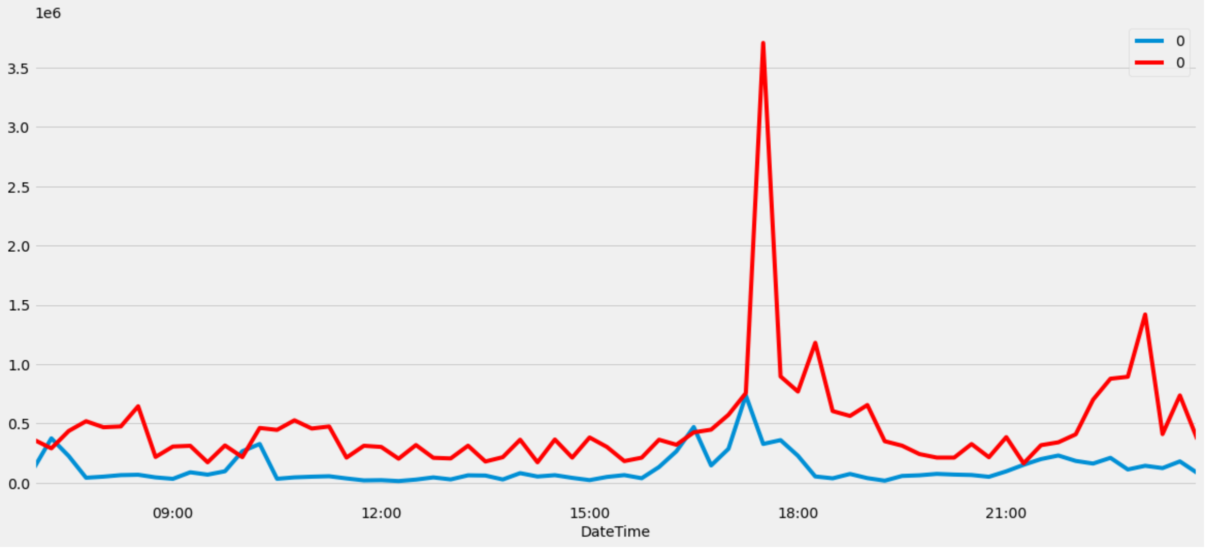
KNN: -3468562.105530 (933187.230664)

RF: -3850315.104455 (1188796.418365)

SVR: -3182063.399199 (1003918.094399)




```
Choosing Linear Resgresion from this -

explained_variance:  -11.2751

mean_squared_log_error:  3.0179

r2:  -19.0031

MAE:  353131.7284

MSE:  313221612142.4934

RMSE:  559662.0517
```



Model Proposed

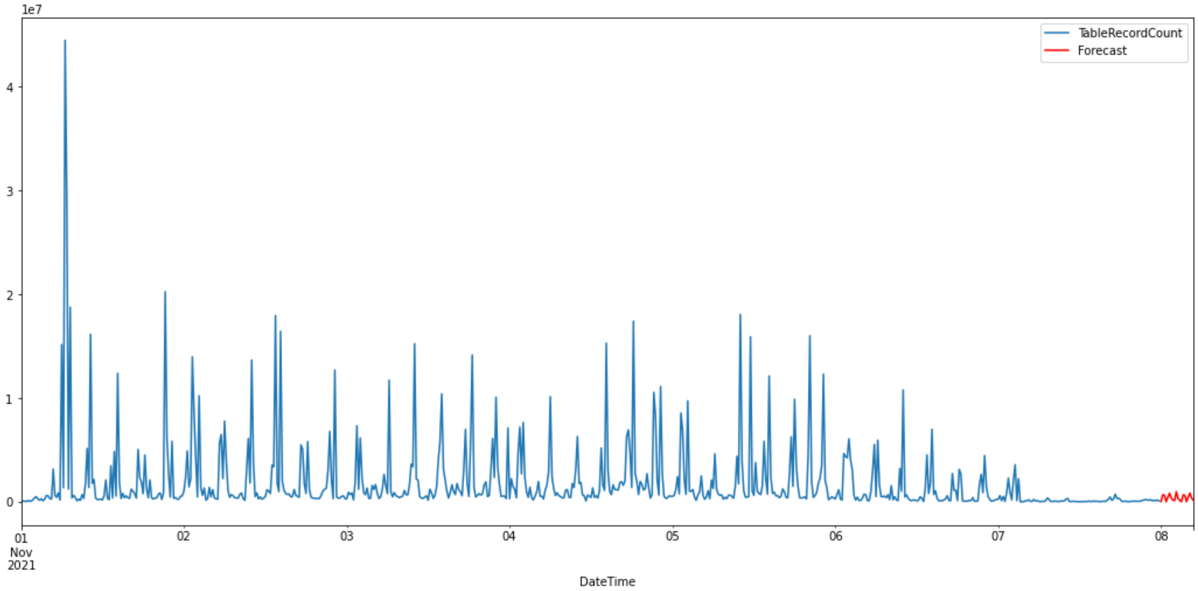
After standardizing the solution with standrad scaler and normalizing the data with normalization using min-max scaler , I split the data.

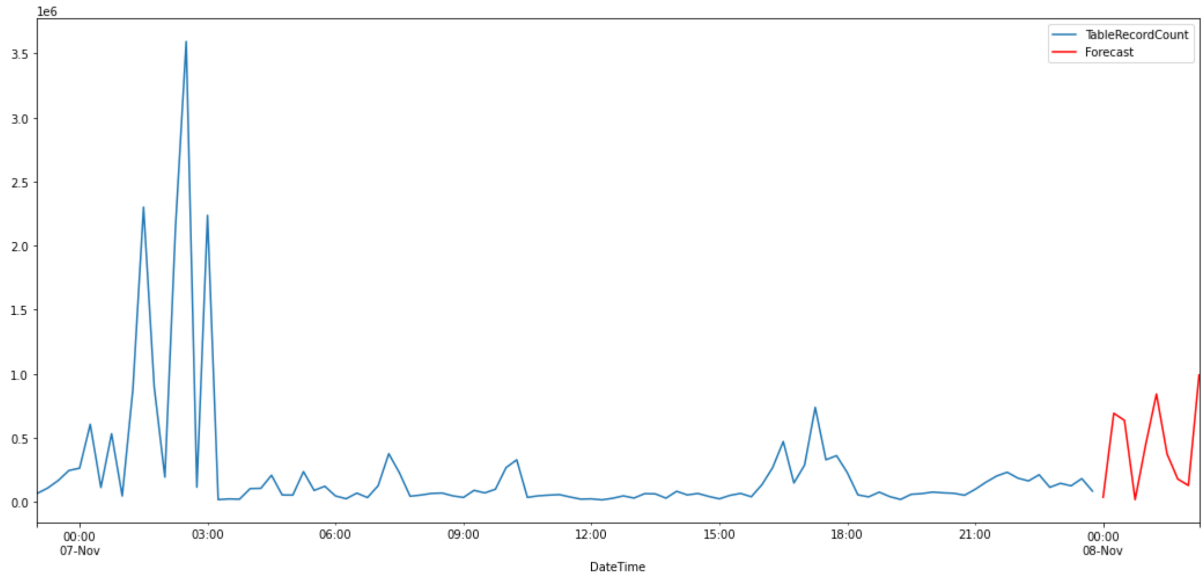
For model I ended up with SARIMAX with hyper - parameters -

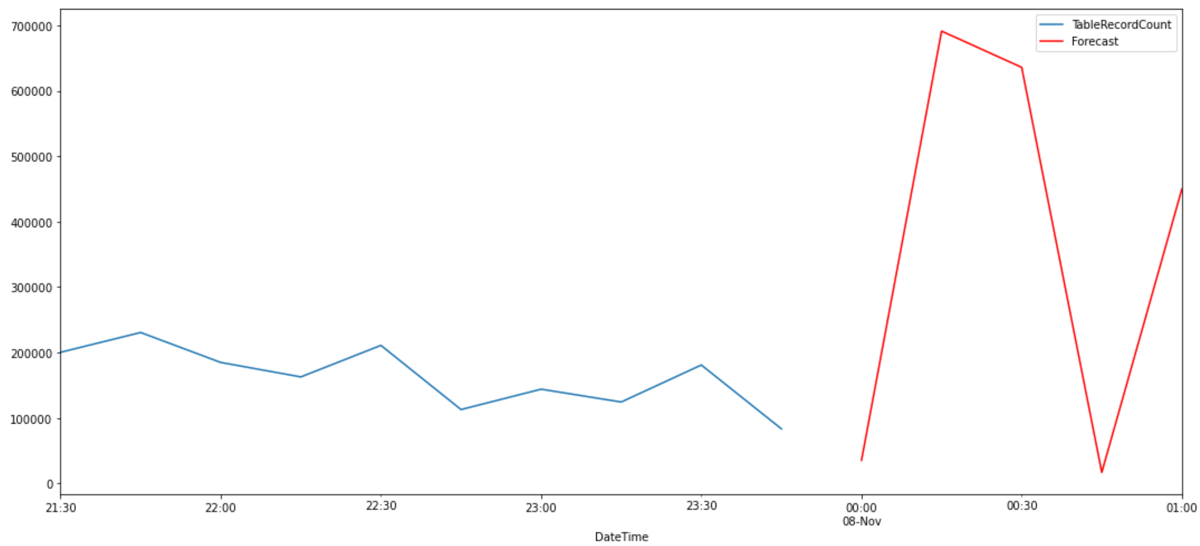
The set of parameters with the minimum AIC is: SARIMA(1, 1, 1)x(0, 1, 1, 12) - AIC:21372.67554496692

On the test data I got a rmse of 0.3316149689811032

Forecasting in the future, this is how the data looks like -







Future Challenges

After this one can create a regression layer that can predict which tabel can get how much records from this total records so that you get more granularity in understadning the tabel records count whereas this solution gives cluster record counts.