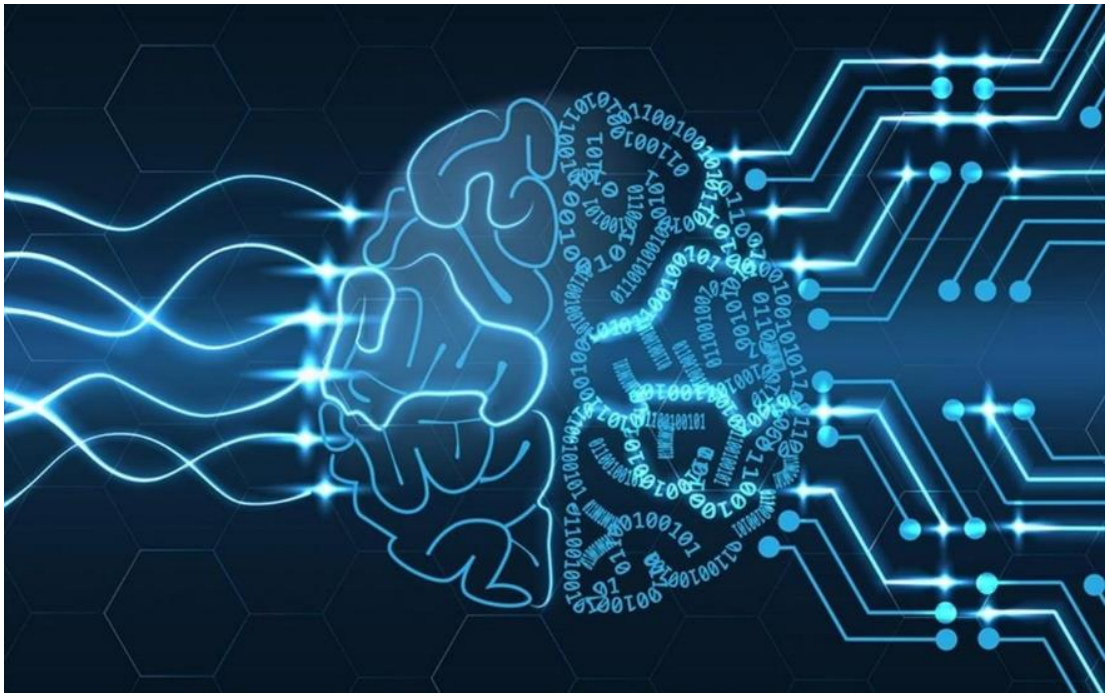


ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ ΚΑΙ ΕΦΑΡΜΟΓΕΣ ΣΤΟΝ ΙΣΤΟ ΤΩΝ ΠΡΑΓΜΑΤΩΝ

ΘΕΟΔΩΡΑ ΑΝΑΣΤΑΣΙΟΥ ΑΜ: ITP19103
ΆΓΓΕΛΟΣ ΛΙΧΑΣ ΑΜ: ITP19123
ΚΩΣΤΑΣ ΜΗΤΣΟΚΑΠΑΣ ΑΜ: ITP19124

Περιεχόμενα

Γενικά για τις εργασίες 2

Εργασία 1 2

Εργασία 2 2

Άσκηση 1 - LSTM Univariate-Multivariate για πρόβλεψη κατανάλωσης ηλεκτρικής ενέργειας 3

Κατανόηση δεδομένων 3

Προετοιμασία δεδομένων - Data Preprocessing 3

Γραφικές Παραστάσεις 4

Προετοιμασία δεδομένων για εισαγωγή στο μοντέλο 9

Παράδειγμα εκτέλεσης 11

Univariate : Global_active_power 11

Multivariate: 'Sub_metering_1','Sub_metering_2','Sub_metering_3' 14

Άσκηση 2 - RNN για αναγνώριση τοξικότητας και διαφόρων υποτυπτών τοξικότητας 18

Κατανόηση δεδομένων 18

Προετοιμασία δεδομένων - Data Preprocessing 20

Σπάσιμο σε Train και Test 21

RNN Model 22

Predict για το test και predict για random Comments απο το test.csv του kaggle 24

AUC - BIAS 25

Γενικά για τις εργασίες

Στις επόμενες σελίδες θα αναλύσουμε τον τρόπο που δουλέψαμε για την υλοποίηση των εργασιών. Οι εργασίες έχουν υλοποιηθεί σε Visual Studio Code και τρέχει τοπικά αφού έγιναν οι απαραίτητες εγκαταστάσεις των εργαλείων tensorflow/keras/pandas/nltk και ότι άλλο χρειάστηκε. Έχουν αναπτυχθεί σε python3.

Στο φάκελο περιέχονται 2 φάκελοι

ΕΡΓΑΣΙΑ 1

Για την συγκεκριμένη εργασία έχουν υλοποιηθεί 3 αρχεία .py για ευκολία

Preparation.py (1)

Visualization.py (2)

multivar_lstm.py (3)

ΕΡΓΑΣΙΑ 2

Για την συγκεκριμένη εργασία έχει υλοποιηθεί ένα αρχείο model_rnn.py.

Κατά την εκτέλεση του αρχείου αυτού δημιουργούνται επιπλέον αρχεία .csv, .h5 για το γράψιμο των δεδομένων και για την αποθήκευση του μοντέλου.

Εντολή εκτέλεσης :

python3 onomaarxeiou.py

Άσκηση 1 - LSTM Univariate-Multivariate για πρόβλεψη κατανάλωσης ηλεκτρικής ενέργειας

ΚΑΤΑΝΟΗΣΗ ΔΕΔΟΜΕΝΩΝ

Αρχικά για την καλύτερη κατανόηση των δεδομένων φτιάξαμε κάποιες γραφικές παραστάσεις ενώ ταυτόχρονα είδαμε ποιο αναλυτικά το αρχείο household_power_consumption.csv τις οποίες θα δούμε σε επόμενη υποενότητα.

```
cols=['Global_active_power','Global_reactive_power',’Voltage’,'Sub_metering_1','Sub_metering_2','Sub_metering_3']
```

ΠΡΟΕΤΟΙΜΑΣΙΑ ΔΕΔΟΜΕΝΩΝ - DATA PREPROCESSING

Είδαμε ότι το αρχείο αυτό περιείχε μια στήλη Date + Time τις οποίες χρειάστηκε να ενωποιήσουμε και να θέσουμε την νέα στήλη DateTime σαν index και datetime, ώστε να μπορούμε να παίζουμε πιο εύκολα με τους χρόνους.

Μετά από αυτό το βήμα γεμίσαμε όλα τα nan με τιμές 0 και μετατρέψαμε τον τύπο της κάθε στήλης σε float. Παράλληλα εντοπίσαμε ότι σε κάποια σημεία το dataset είχε ‘?’ αντι για κάποια τιμή και έτσι τα αντικαταστήσαμε και αυτά με 0.

(- μπορούσε να γίνει και σαν mean των 10 τελευταιων - try)

```
df['Global_active_power'].fillna(0.0, inplace=True)
df['Global_reactive_power'].fillna(0.0, inplace=True)
df['Voltage'].fillna(0.0, inplace=True)
df['Global_intensity'].fillna(0.0, inplace=True)
df['Sub_metering_1'].fillna(0.0, inplace=True)
df['Sub_metering_2'].fillna(0.0, inplace=True)
df["Sub_metering_3"].fillna(0.0, inplace=True)
```

Αργότερα χρησιμοποιώντας τον Grouper απο pandas, κάναμε τα δεδομένα groupby ανα 12ώρα, και τις στήλες τις γεμίσαμε με το .sum() τις συγκεκριμένης ώρας. (κάθε ώρα ήταν αρχικά χωρισμένη σε λεπτά έτσι παίρνει το sum() των 720 λεπτών και το βάζει σαν νέα τιμή στην αντίστοιχη θέση.)

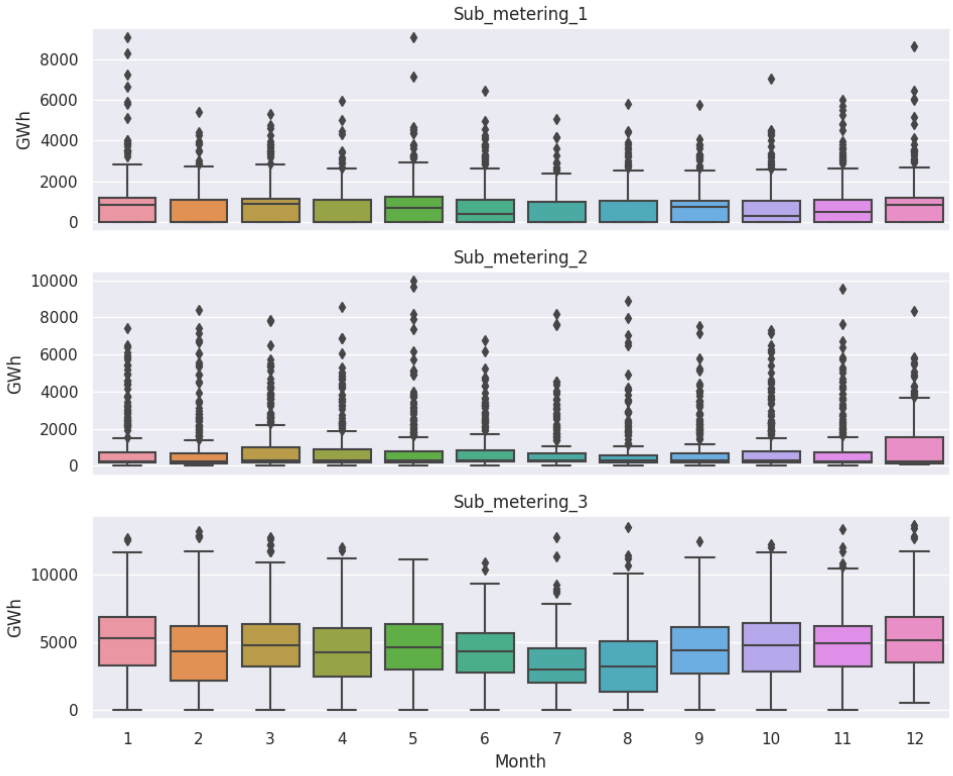
```
df=df.groupby(pd.Grouper(freq='12H'))['Global_active_power','Global_reactive_power','Voltage','Global_intensity','Sub_metering_1','Sub_metering_2','Sub_metering_3'].sum()
```

Έγιναν δοκιμές και με group 15Min, 1H, 3Months, αλλά εν τέλη κρατήσαμε το ανα ώρα.

Στην συνέχεια φτιάξαμε κάποιες επιπλέον στήλες που μας βοήθησαν στην καλύτερη οπτικοποίηση των δεδομένων .

```
df['Month'] = df.index.month #find week based on datetime
df['Quarter'] = df.index.quarter #find week based on datetime
df['Year'] = df.index.year #find week based on datetime
df['Day'] = df.index.weekday #find week based on datetime
```

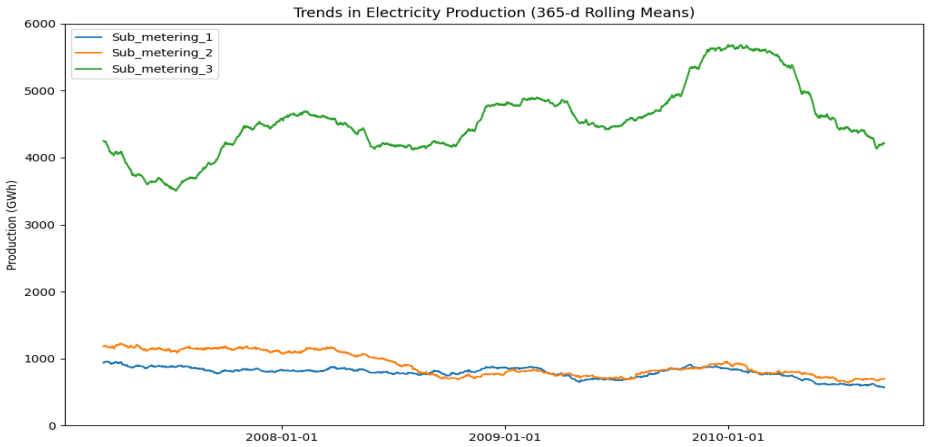
ΓΡΑΦΙΚΕΣ ΠΑΡΑΣΤΑΣΕΙΣ



Εικόνα 1

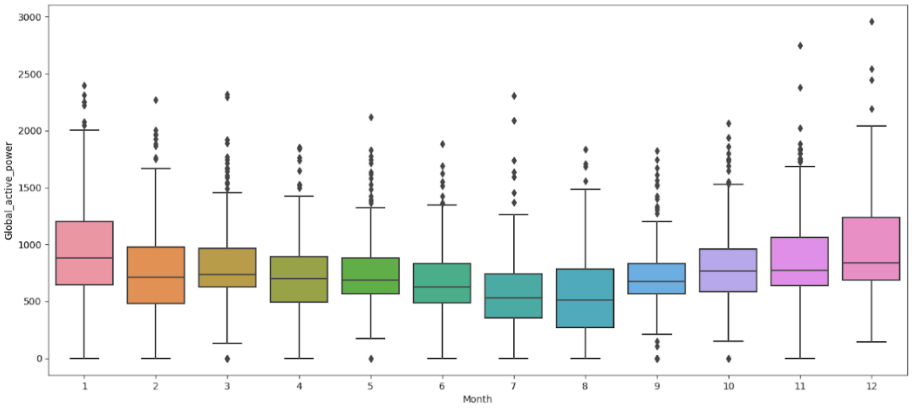
Στην εικόνα 1, βλέπουμε ότι το sub_metering_2 που αναφέρεται σε laundry room, containing a washing-machine, a tumble-drier, a refrigerator and a light έχει μεγαλύτερη χρήση τον Δεκέμβριο είτε λόγο πιο χαμηλών θερμοκρασιών είτε λόγο εορτών για το φώς.

Ενώ το sub_metering_3 περισσότερο τους χειμερινούς μήνες electric water-heater and an air-conditioner .



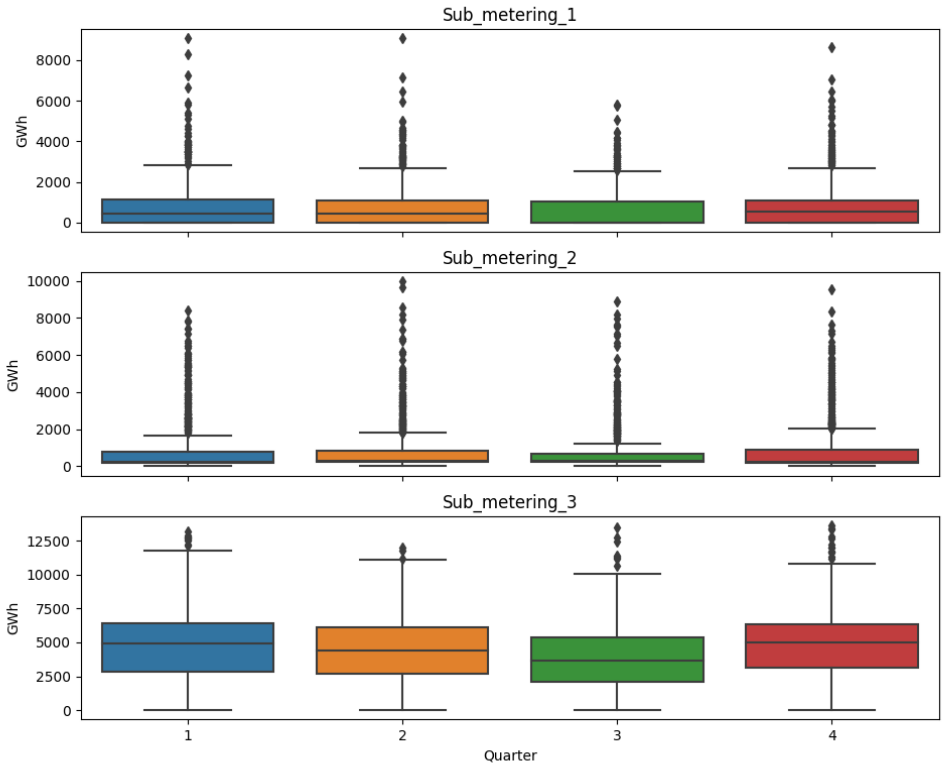
Εικόνα 2

Στην εικόνα 2 βλέπουμε ότι το Sub_metering_3 το οποίο αντιστοιχεί στην **** αυξάνεται κατα τους χειμερινούς μήνες. Ελάχιστη αύξηση παρατηρούμε και στα άλλα δύο sub_metering_2 & sub_metering_1



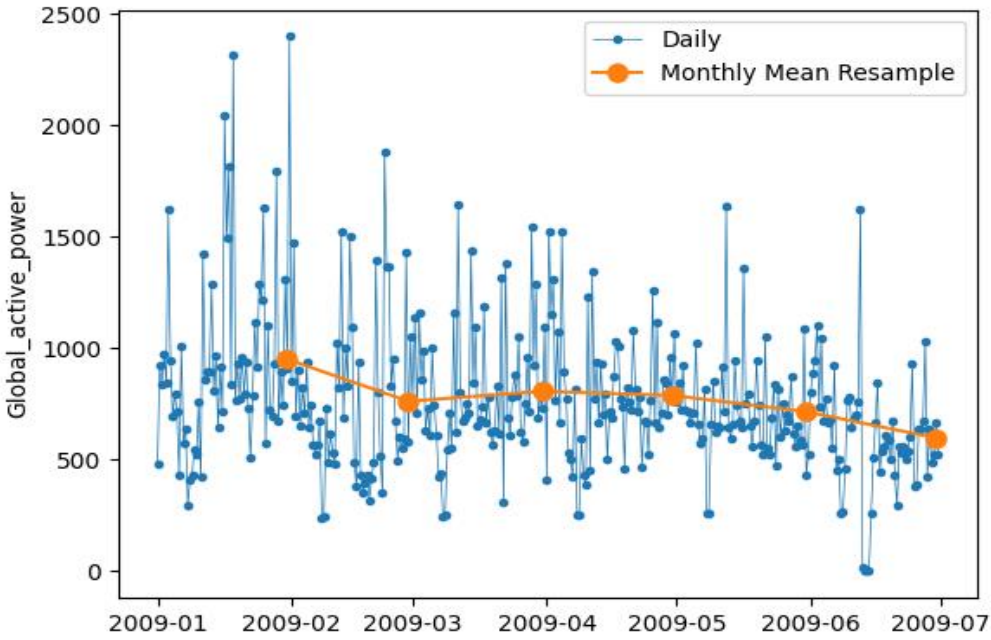
Εικόνα 3

Στην εικόνα 3 βλέπουμε ότι οι μήνες Ιανουάριος και Δεκέμβριος έχουν περισσότερη κατανάλωση απο τους υπόλοιπους μήνες, ενώ οι καλοκαιρινοί μήνες πιό χαμηλή κατανάλωση.

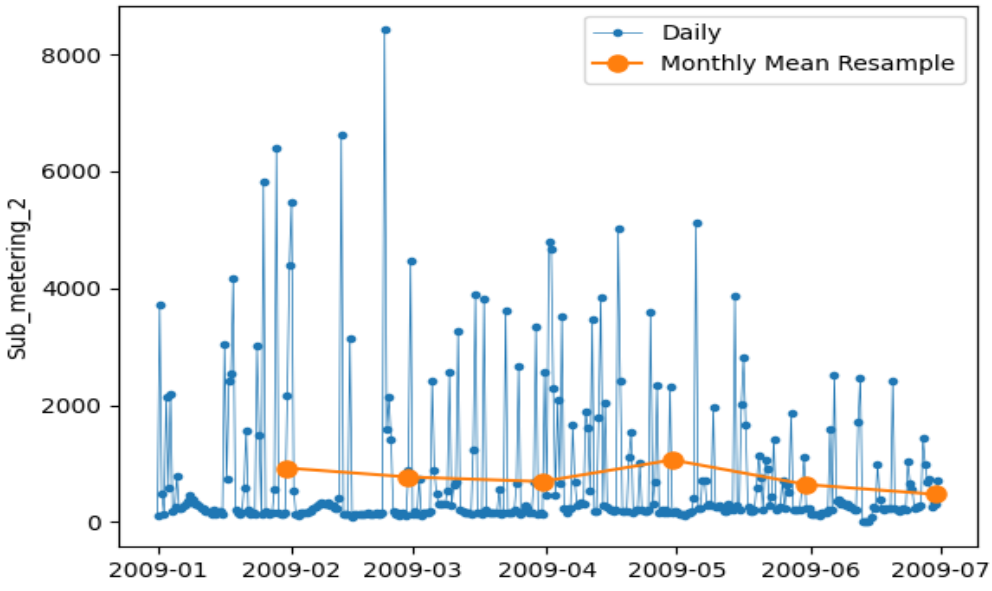


Εικόνα 4

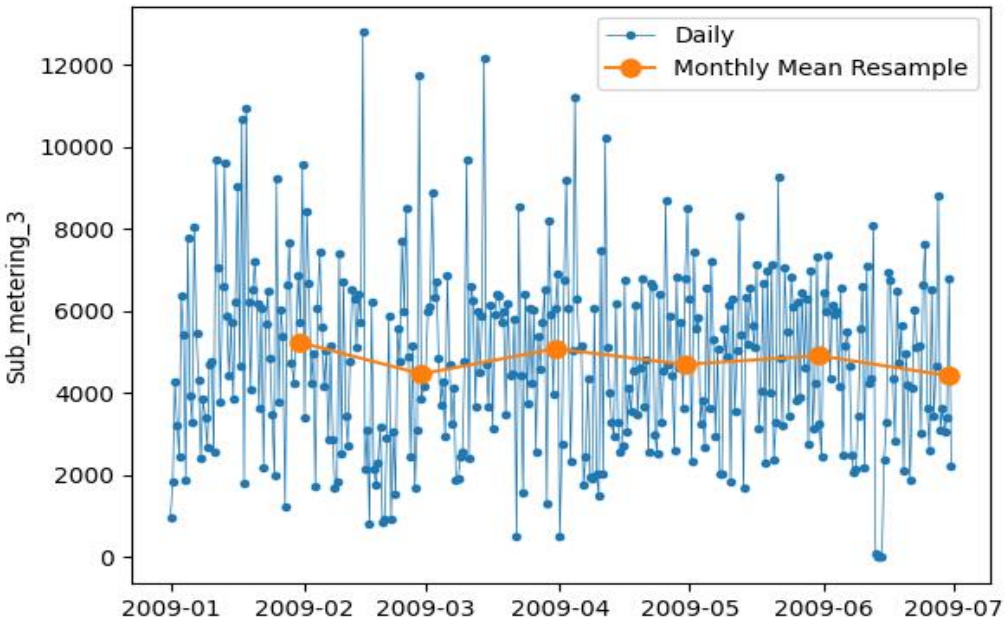
Στην εικόνα 4 βλέπουμε ότι στο sub_metering_3 που αναφέρεται σε θερμοσίφωνα και a/c είναι ελάχιστα πιο ψηλά απο τα υπόλοιπα Quarter που έχουν υψηλότερες θερμοκρασίες.



Εικόνα 5

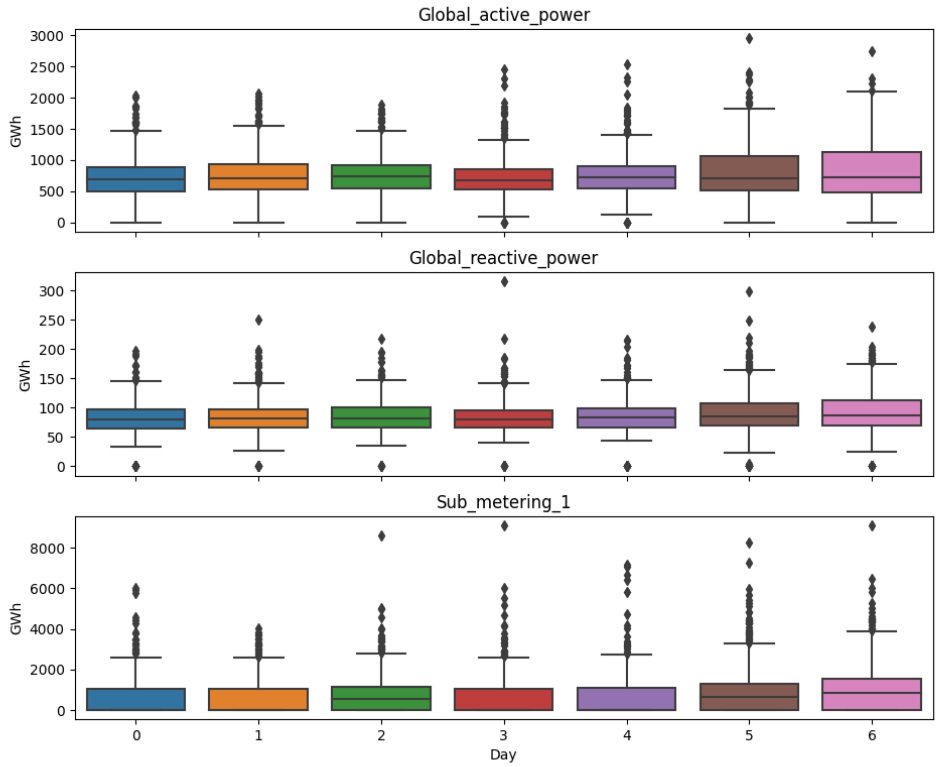


Εικόνα 6



Εικόνα 7

Εικόνα 8



Εικόνα 9

Στην εικόνα 10 βλέπουμε ότι τα Σαββατοκύριακα η κατανάλωση ηλεκτρικής ενέργειας(Global_active_power) αυξάνεται και αυτό είναι αναμενόμενο αφού οι περισσότεροι δεν δουλεύουν.

ΠΡΟΕΤΟΙΜΑΣΙΑ ΔΕΔΟΜΕΝΩΝ ΓΙΑ ΕΙΣΑΓΩΓΗ ΣΤΟ ΜΟΝΤΕΛΟ

Έχουμε φτιάξει ένα μοντέλο το οποίο υποστηρίζει και univariate και multivariate δεδομένα. Ανάλογα με το αν το len(features_considered) δηλαδή με το πόσες στήλες δώσαμε συνεχίζει ανάλογα ώστε να φτιάξει σωστά τις περιοχές που θα χρησιμοποιήσει για την εκπαίδευση και τις περιοχές που θα χρησιμοποιήσει για προβλεψη, ενώ επίσης χρειάζεται να ορίζουμε και τον αριθμό των STEPS για να φτιαχτούν σωστά τα δεδομένα μας. Χρησιμοποιήθηκαν STEPS = 1,6,12 για τις εκτελέσεις μας.

Πριν την εισαγωγή των δεδομένων στην συνάρτηση “δημιουργίας” των δεδομένων, χρησιμοποιήσαμε τον MinMaxScaler για να κανονικοποιήσουμε τα δεδομένα μας.

```
scaler = MinMaxScaler(feature_range=(0, 1)) #normalize ta dedomena
me xrisi tu minmaxscaler
dataset = scaler.fit_transform(features)
```

Στην συνέχεια αφού τα δεδομένα σπάσουν σε ομάδες και αφού ορίσουμε την περιοχή εκπαίδευσης και πρόβλεψης , χρειάζεται να γίνει κάποια cache() των δεδομένων.

```
x_train_multi, y_train_multi = multivariate_data(dataset, data ,
past_history,future_target, STEP)
x_val_multi, y_val_multi = multivariate_data(dataset,
data,  past_history, future_target, STEP)

#cache() & repeat gia ligoteri mnimi - amesi prosvasi
train_data_multi =
tf.data.Dataset.from_tensor_slices((x_train_multi,
y_train_multi))
train_data_multi=train_data_multi.cache().shuffle(BUFFER_SIZE).ba
tch(BATCH_SIZE).repeat()
val_data_multi = tf.data.Dataset.from_tensor_slices((x_val_multi,
y_val_multi))
val_data_multi = val_data_multi.batch(BATCH_SIZE).repeat()
```

Το μοντέλο μας αποτελείται από

- Sequential
- 2 LSTM το ένα εκ των οποίων έχει εσωτερικό dropout.

```
#MODEL
multi_step_model = tf.keras.models.Sequential()
multi_step_model.add(tf.keras.layers.LSTM(16,
return_sequences=True,                                  input_sha
pe=x_train_multi.shape[-2:]))
multi_step_model.add(tf.keras.layers.LSTM(16,
activation='relu',dropout=0.2))
```

Στη συνέχεια το μοντέλο γίνεται compile με την χρήση SGD Optimizer και learning rate = 0.01. Ενώ στην συνέχεια γίνεται το fit στο οποίο ορίζονται οι εποχές, το validation_data τα validation_steps, steps_per_epoch.

```
#compile model with SGD Optimizer
multi_step_model.compile(optimizer=tf.keras.optimizers.SGD(
    learning_rate=0.01), loss='mae')

multi_step_history = multi_step_model.fit(train_data_multi,
epochs=EPOCHS, steps_per_epoch=20,validation_data=val_data_multi,
validation_steps=10,verbose=1)
```

Μετά την εκπαίδευση του μοντέλου το χρησιμοποιούμε για να κάνει κάποια predictions .

Και εν τέλη παρουσιάζουμε τα αποτελέσματα σε γραφική παράσταση χρησιμοποιώντας τις πραγματικές τιμές και τις τιμές τις οποίες πρόβλεψε το μοντέλο μας.

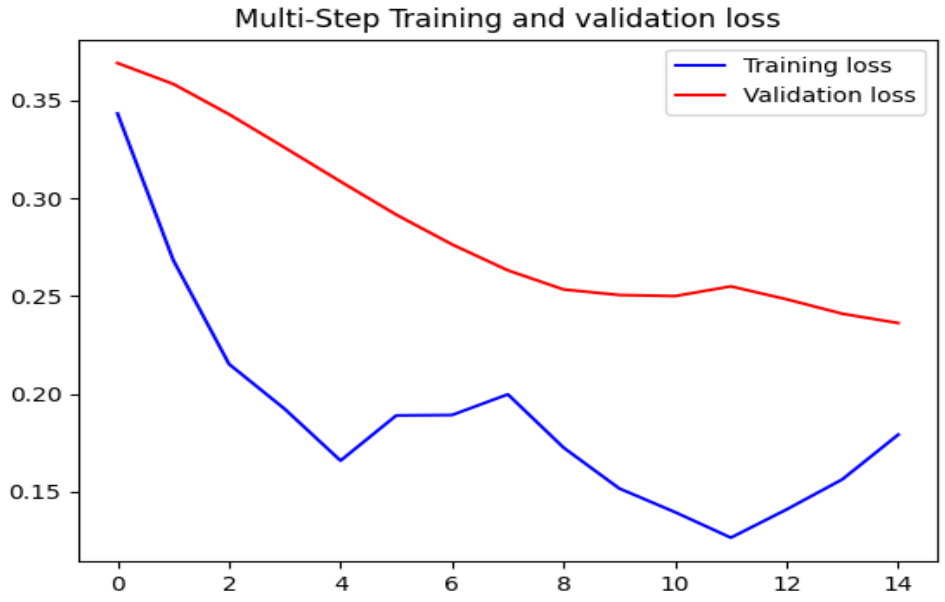
ΠΑΡΑΔΕΙΓΜΑ ΕΚΤΕΛΕΣΗΣ

Univariate : Global_active_power

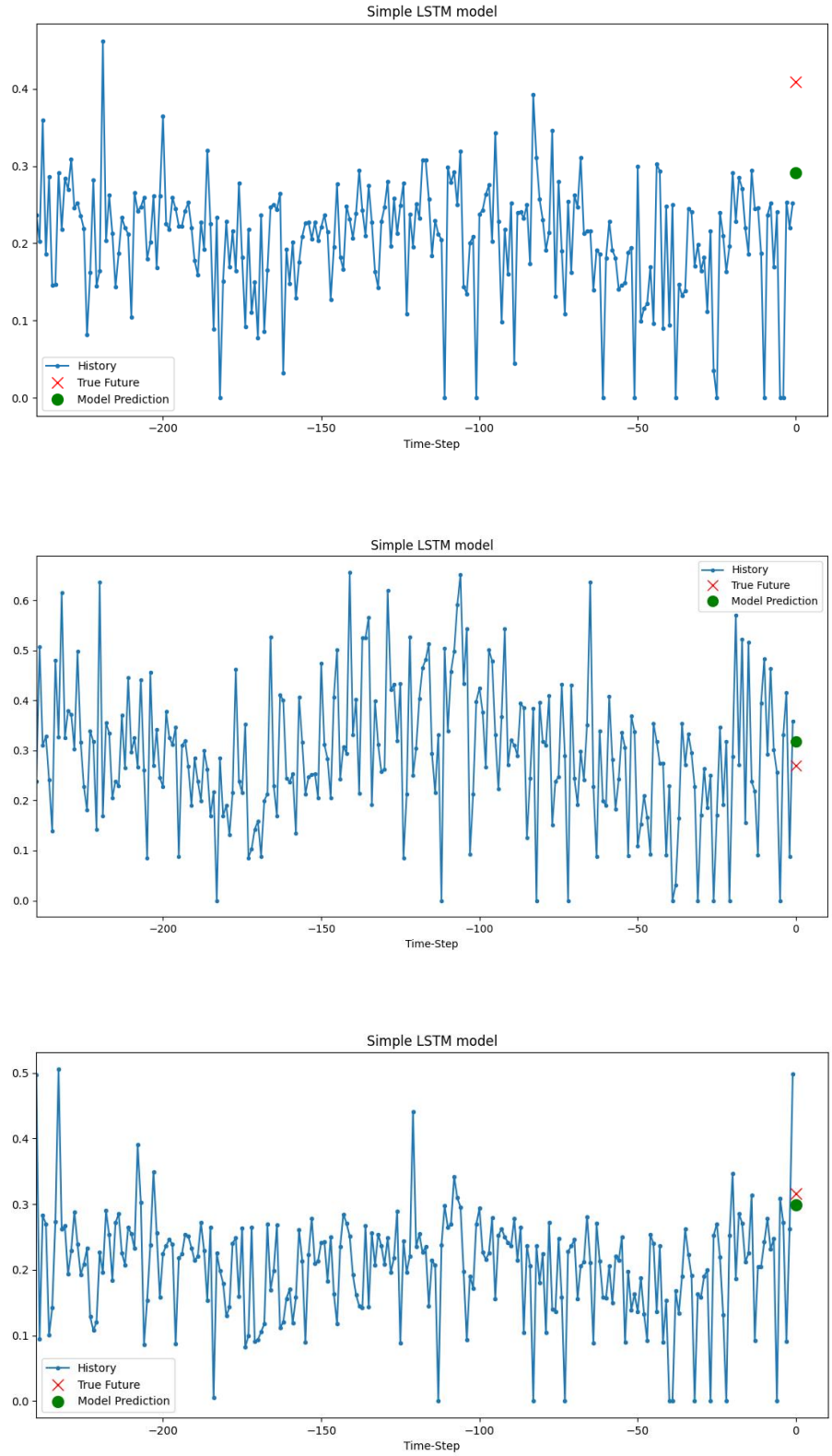
```
past_history =  1440 #arxika 1440 grammes ara peripou 24 mines - 2
xronia
STEP = 6
EPOCHS = 15 #epochs
EVALUATION_INTERVAL = 40
future_target = 1 #auta pou thelw n predict meta - tosa samples
BATCH_SIZE = 5 #batch size g to training
BUFFER_SIZE = 100 #buffer size g tin cache()
```

```
Epoch 1/15
20/20 [==============================] - 2s 82ms/step - loss: 0.2059 - val_loss: 0.1631
Epoch 2/15
20/20 [==============================] - 1s 66ms/step - loss: 0.1320 - val_loss: 0.1924
Epoch 3/15
20/20 [==============================] - 1s 68ms/step - loss: 0.1028 - val_loss: 0.1911
Epoch 4/15
20/20 [==============================] - 1s 68ms/step - loss: 0.1047 - val_loss: 0.2365
Epoch 5/15
20/20 [==============================] - 1s 69ms/step - loss: 0.0897 - val_loss: 0.1992
Epoch 6/15
20/20 [==============================] - 1s 68ms/step - loss: 0.1080 - val_loss: 0.1914
Epoch 7/15
20/20 [==============================] - 1s 69ms/step - loss: 0.1026 - val_loss: 0.1668
Epoch 8/15
20/20 [==============================] - 1s 70ms/step - loss: 0.1127 - val_loss: 0.1802
Epoch 9/15
20/20 [==============================] - 1s 71ms/step - loss: 0.0932 - val_loss: 0.1710
Epoch 10/15
20/20 [==============================] - 1s 69ms/step - loss: 0.0932 - val_loss: 0.1966
Epoch 11/15
20/20 [==============================] - 1s 69ms/step - loss: 0.0842 - val_loss: 0.2110
Epoch 12/15
20/20 [==============================] - 1s 68ms/step - loss: 0.0956 - val_loss: 0.2211
Epoch 13/15
20/20 [==============================] - 1s 69ms/step - loss: 0.0951 - val_loss: 0.1677
Epoch 14/15
20/20 [==============================] - 1s 68ms/step - loss: 0.0991 - val_loss: 0.1683
Epoch 15/15
20/20 [==============================] - 1s 69ms/step - loss: 0.1189 - val_loss: 0.1657
univariateplot
```

Εικόνα 10



Εικόνα 11

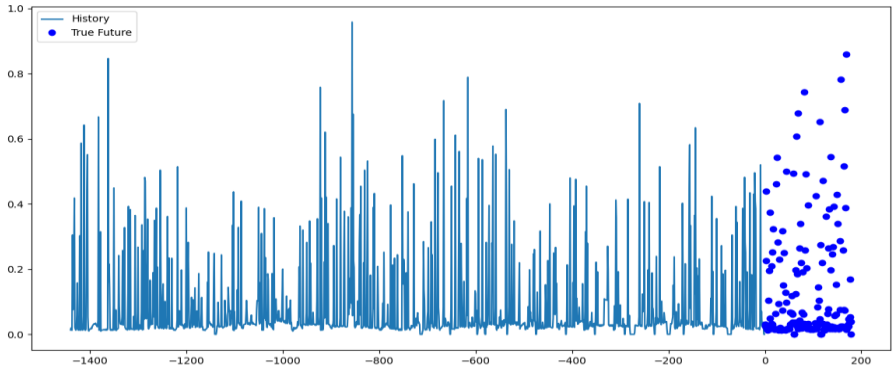

Simple LSTM model
History
True Future
Model Prediction
Time-Step
Simple LSTM model
History
True Future
Model Prediction
Time-Step
Simple LSTM model
History
True Future
Model Prediction
Time-Step

Εικόνα 12

Multivariate: 'Sub_metering_1','Sub_metering_2','Sub_metering_3'

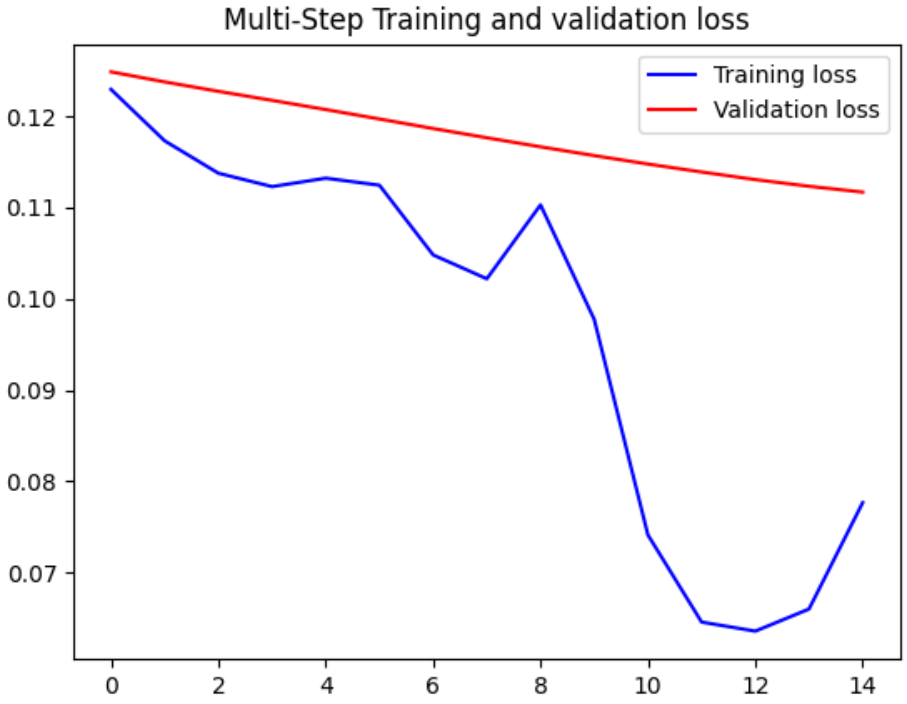
Παράδειγμα 1.

```
past_history =  1440 #arxika 51855 grammes ara peripou 36 mines
STEP = 1
EPOCHS = 15 #epochs
EVALUATION_INTERVAL = 40
future_target = 180 #auta pou thelw n predict meta - tosa samples
BATCH_SIZE = 5 #batch size g to training
BUFFER_SIZE = 100 #buffer size g tin cache()
```

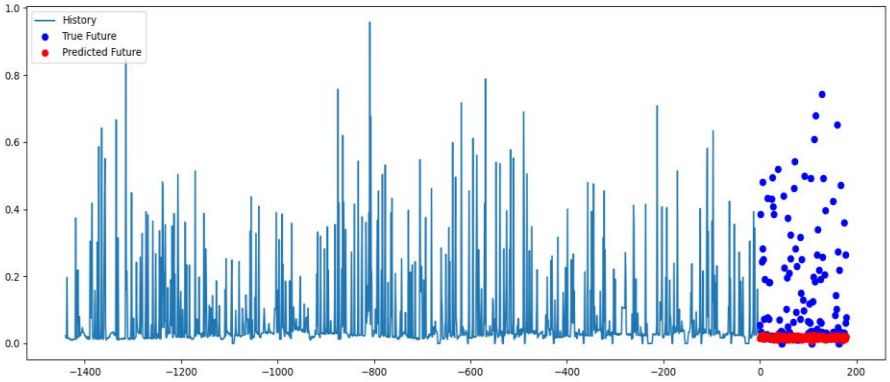


Εικόνα 13

Θέλω να μαντέψω 180 samples - δηλαδη 3 μήνες



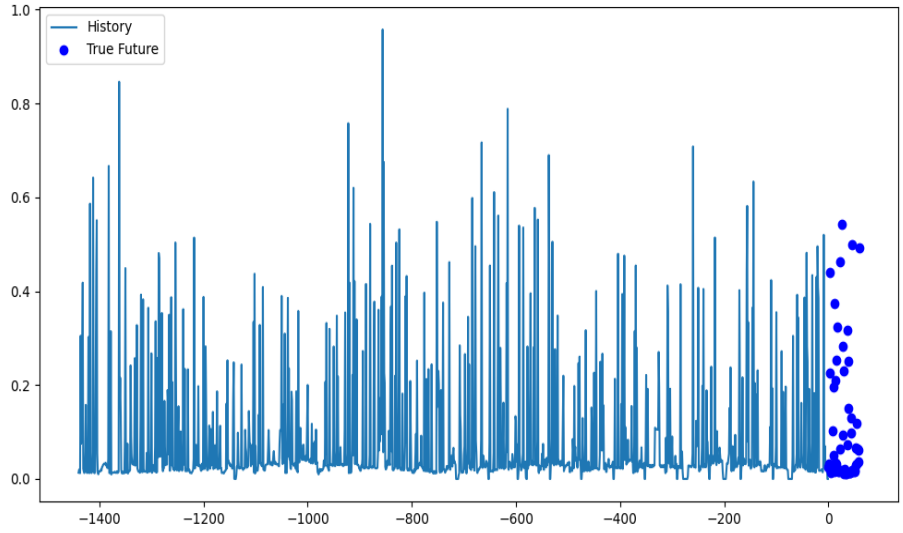
Εικόνα 14 Γραφική Train και Loss για το συγκεκριμένο παράδειγμα



Εικόνα 15 Το αποτέλεσμα τελικά που προέβλεψε το μοντέλο μας.

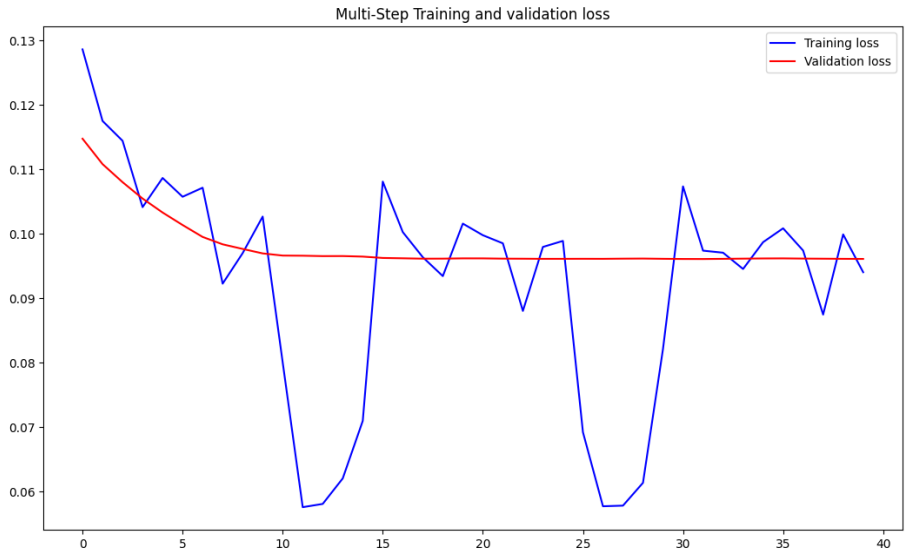
Παράδειγμα 2.

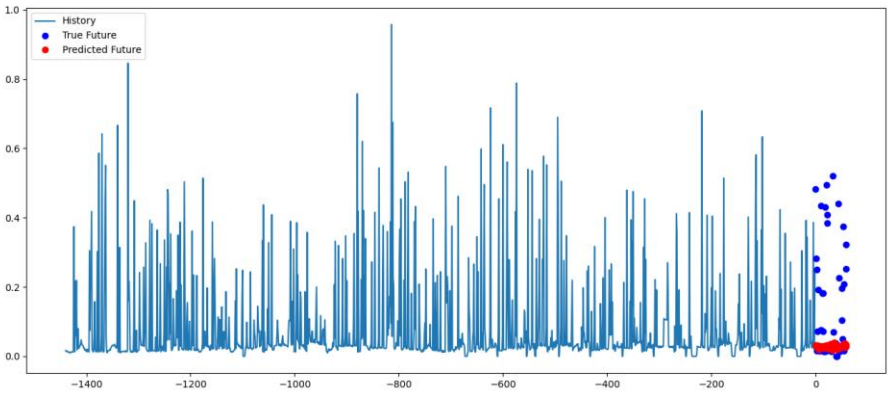
```
past_history =  1440 #arxika 51855 grammes ara peripou 36 mines
STEP = 1
EPOCHS = 40 #epochs
EVALUATION_INTERVAL = 40
future_target = 60 #auta pou thelw n predict meta - tosa samples
BATCH_SIZE = 5 #batch size g to training
BUFFER_SIZE = 100 #buffer size g tin cache()
```



Θέλω να μαντέψω 60 samples - δηλαδη 1 μήνα

```
20/20 [==============================] - 12s 586ms/step - loss: 0.0577 - val_loss: 0.0961
Epoch 28/40
20/20 [==============================] - 12s 594ms/step - loss: 0.0578 - val_loss: 0.0961
Epoch 29/40
20/20 [==============================] - 12s 609ms/step - loss: 0.0614 - val_loss: 0.0961
Epoch 30/40
20/20 [==============================] - 12s 605ms/step - loss: 0.0821 - val_loss: 0.0961
Epoch 31/40
20/20 [==============================] - 13s 664ms/step - loss: 0.1073 - val_loss: 0.0961
Epoch 32/40
20/20 [==============================] - 12s 599ms/step - loss: 0.0974 - val_loss: 0.0961
Epoch 33/40
20/20 [==============================] - 12s 583ms/step - loss: 0.0971 - val_loss: 0.0961
Epoch 34/40
20/20 [==============================] - 11s 570ms/step - loss: 0.0945 - val_loss: 0.0961
Epoch 35/40
20/20 [==============================] - 11s 562ms/step - loss: 0.0987 - val_loss: 0.0962
Epoch 36/40
20/20 [==============================] - 13s 632ms/step - loss: 0.1008 - val_loss: 0.0962
Epoch 37/40
20/20 [==============================] - 12s 599ms/step - loss: 0.0974 - val_loss: 0.0961
Epoch 38/40
20/20 [==============================] - 12s 609ms/step - loss: 0.0874 - val_loss: 0.0961
Epoch 39/40
20/20 [==============================] - 13s 631ms/step - loss: 0.0999 - val_loss: 0.0961
Epoch 40/40
20/20 [==============================] - 12s 606ms/step - loss: 0.0940 - val_loss: 0.0961
```





Το αποτέλεσμα τελικά που προέβλεψε το μοντέλο μας.

Άσκηση 2 - RNN για αναγνώριση τοξικότητας και διαφόρων υποτυπτών τοξικότητας.

ΚΑΤΑΝΟΗΣΗ ΔΕΔΟΜΕΝΩΝ

Αρχικά για την καλύτερη κατανόηση των δεδομένων φτιάξαμε κάποιες γραφικές παραστάσεις ενώ ταυτόχρονα είδαμε ποιο αναλυτικά το αρχείο train.csv το οποίο και χρησιμοποιήσαμε για το train και test κομματι που θα αναλύσουμε αργότερα σε αυτή την αναφορά. Στα αρχικά στάδια της εργασίας χρησιμοποιούσαμε μικρότερα κομμάτια από τα δεδομένα του train.csv ενω τα τελικά μας αποτελέσματα προέκυψαν από την ανάλυση ολόκληρου το dataset.

```
7.77% of all comments are toxic
2 comments refer to the severe_toxicity identity, 100.00% are toxic
5006 comments refer to the obscene identity, 98.04% are toxic
6125 comments refer to the identity_attack identity, 92.73% are toxic
2095 comments refer to the threat identity, 97.37% are toxic
21609 comments refer to the male identity, 14.94% are toxic
27912 comments refer to the female identity, 13.91% are toxic
895 comments refer to the transgender identity, 22.12% are toxic
5 comments refer to the other_gender identity, 60.00% are toxic
525 comments refer to the heterosexual identity, 22.10% are toxic
4780 comments refer to the homosexual_gay_or_lesbian identity, 28.08% are toxic
126 comments refer to the bisexual identity, 18.25% are toxic
4 comments refer to the other_sexual_orientation identity, 0.00% are toxic
21586 comments refer to the christian identity, 8.50% are toxic
4479 comments refer to the jewish identity, 15.20% are toxic
12366 comments refer to the muslim identity, 22.15% are toxic
323 comments refer to the hindu identity, 10.22% are toxic
257 comments refer to the buddhist identity, 8.17% are toxic
652 comments refer to the atheist identity, 11.81% are toxic
164 comments refer to the other_religion identity, 15.85% are toxic
6240 comments refer to the black identity, 29.55% are toxic
9858 comments refer to the white identity, 27.62% are toxic
2263 comments refer to the asian identity, 12.20% are toxic
980 comments refer to the latino identity, 18.47% are toxic
219 comments refer to the other_race_or_ethnicity identity, 17.81% are toxic
51 comments refer to the physical_disability identity, 9.80% are toxic
44 comments refer to the intellectual_or_learning_disability identity, 11.36% are toxic
2577 comments refer to the psychiatric_or_mental_illness identity, 19.17% are toxic
2 comments refer to the other_disability identity, 50.00% are toxic
2353 comments refer to the sexual_explicit identity, 87.55% are toxic
Found 921102 unique tokens.
```

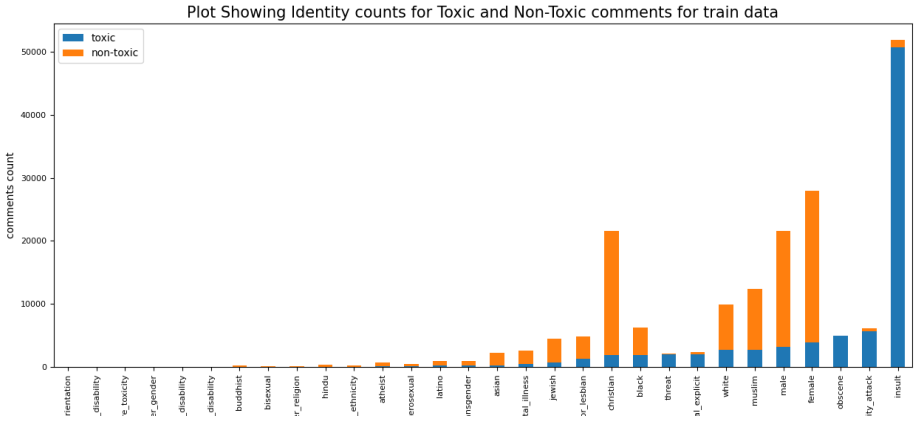
Εικόνα 1

Στην εικόνα 1 βλέπουμε τα ποσοστά τοξικότητας για ολόκληρο το dataset (1804874 comments). Γενικά βλέπουμε ότι στο συγκεκριμένο σύνολο δεδομένων η τοξικότητα ανιχνεύεται στο 7.77% των σχολίων, ένα σχετικά μικρό ποσοστό που εν τέλη δεν δυσκόλεψε το μοντέλο στην ανίχνευση των “κακών” λέξεων ώστε να εξάγει αργότερα τις σωστές προβλέψεις.

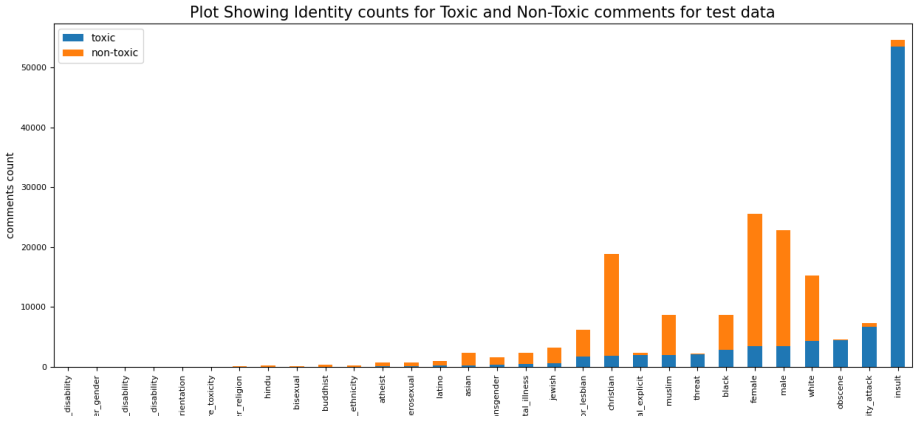
Στην εικόνα 1 μπορούμε επίσης να δούμε αναλυτικά τα ποσοστά τοξικότητας για κάθε identity ξεχωριστά. Έτσι για παράδειγμα βλέπουμε ότι για την ομάδα “sexual_explicit_identity” υπάρχουν αναφορές σε 2353 σχόλια, και από αυτά το 87.55% είναι τοξικό. Αντίστοιχα μπορούν να αναλυθούν τα δεδομένα και για τις υπόλοιπες ομάδες.

(*Στην τελευταία γραμμή της εικόνας 1, βλέπουμε τον αριθμό (921102) των ξεχωριστών λέξεων που έσπασαν τα σχόλια μετά την χρήση του tokenizer)

Παράλληλα στα σχήματα 1&2 μπορούμε να δούμε οπτικοποιημένα τα ποσοστά τοξικών σχολίων μετά το σπάσιμο του αρχικού dataset σε train/test .



Σχήμα 1



Σχήμα 2

ΠΡΟΕΤΟΙΜΑΣΙΑ ΔΕΔΟΜΕΝΩΝ - DATA PREPROCESSING

Μετά την κατανόηση των δεδομένων ήμασταν σε θέση να τα επεξεργαστούμε και να τα καθαρίσουμε ώστε να βοηθήσουμε το μοντέλο μας να εξάγει πιο accurate προβλέψεις.

Κατα το καθάρισμα των δεδομένων ουσιαστικά :

- Κάναμε drop στήλες που θεωρήσαμε επιπλέον π.χ

```
'created_date','publication_id','parent_id','article_id','ra
ting','funny','wow','sad','likes','disagree','identity_annot
ator_count','toxicity_annotator_
```

- Στη συνέχεια για την στήλη 'Target' είπαμε ότι όταν είναι >=0,5 τότε θα ήταν TRUE - toxic ενώ εάν ήταν <0.5 τότε θα ήταν FALSE-non toxic.
- Κάτι αντίστοιχο ακολουθήσαμε και για τις υπόλοιπες στήλες ώστε εν τέλη να έχουμε στήλες μόνο με τιμές boolean

```
identity_columns = [
'severe_toxicity','obscene','identity_attack','threat',
'male', 'female', 'transgender', 'other_gender',
'heterosexual', 'homosexual_gay_or_lesbian', 'bisexual',
'other_sexual_orientation', 'christian', 'jewish', 'muslim',
'hindu', 'buddhist', 'atheist', 'other_religion', 'black',
'white', 'asian', 'latino', 'other_race_or_ethnicity',
'physical_disability',
'intellectual_or_learning_disability','psychiatric_or_mental
_illness', 'other_disability','sexual_explicit']
```

- Καθαρίσαμε απο την στήλη "Comment_Text" τις αγκύλες, τους "κακούς" χαρακτήρες, τους αριθμούς, τα http/www και τα αντικαταστήσαμε με κενούς χαρακτήρες . Αργότερα μετατρέψαμε όλους τους χαρακτήρες σε μικρούς - lower, τα σπάσαμε σε tokens με χρήση του tokenizer, ενώ αργότερα χρησιμοποιώντας το stopwords της nltk ξεχωρίσαμε τις σημαντικές λέξεις που θα κρατούσαμε για κάθε σχόλιο στην συγκεκριμένη στήλη.

```
REPLACE_SLASH = re.compile('[/(){\}[\]\|@,;]')
BAD_SYMBOLS = re.compile('[^0-9a-z #+_]')
STOPWORDS = set(stopwords.words('english'))
NUMERIC = re.compile('123456789')
HTTP = re.compile('http\S+')
WWW = re.compile('www\S+')
H1 = re.compile('\b\w{1}\b')
AD_SPACES = re.compile(' +')
```

- Στη συνέχεια χρησιμοποιήσαμε την fit_on_texts του tokenizer αλλά και την pad sequences ώστε οι σειρές από λέξεις να έχουν εν τέλη το ιδιο length ώστε να μπορεί να τις επεξεργαστεί αργότερα το νευρωνικό.

ΣΠΑΣΙΜΟ ΣΕ TRAIN ΚΑΙ TEST

Αργότερα χρειάστηκε να σπάσουμε το μεγάλο αρχείο σε train-test , ενω παράλληλα έπρεπε να ξεχωρίσουμε τις στήλες που θα χρησιμοποιούσε το μοντέλο σαν target στήλη (δηλαλή αυτή την οποία περιμέναμε να μας προβλέψει) ενώ αντίστοιχα να χρησιμοποιήσει την comment_text για να μας βγάλει το αποτέλεσμα.

Η στήλη target χρειάστηκε να μετατραπεί σε πίνακα dummies, ενώ η στήλη comment_Text με padding ώστε το κάθε σχόλιο να είχε το ίδιο length με κάποιο άλλο, όπως αναφέρθηκε και παραπάνω.

```
X =
tokenizer.texts_to_sequences(train_dataset['comment_text'].values
)# Padding the sequences - for equal comment length.
X = pad_sequences(X, maxlen=MAX_SEQUENCE_LENGTH) #gia na exun to
idio length ta sequences mas
Y = pd.get_dummies(train_dataset['target']) #dummies
```

Στην συνέχεια χρησιμοποιούμε το 90% του train που σπάσαμε παραπάνω για training και το υπόλοιπο 10% για validation. Το test το χρησιμοποιήσαμε αργότερα για τον υπολογισμό και την ανάλυση του bias.

```
X_train, X_test, Y_train, Y_test =
train_test_split(np.array(X),np.array(Y), test_size = 0.10,
random_state = 33)
```

RNN MODEL

Το μοντέλο μας αποτελείται από :

- ένα Embedding Layer για να μαθαίνει ένα embedding για κάθε λέξη στο training-set.
- ενα Spatial Dropout 1D
- ενα Bidirectional LSTM 100 layer
- ένα Dense Layer

Στην συνέχεια το μοντέλο μας γίνεται compile με χρήση του RMSprop optimizer, με learning rate = 0.001 , παρόλα αυτά έγιναν δοκιμές και με άλλες τιμές learning rate και εν τέλη κρατήσαμε αυτή την τιμή, σαν loss χρησιμοποιούμε την Binary_Crossentropy αφού χρειάζεται να μαντέψει τιμές 0 ή 1. (Target = TRUE/FALSE).

Το μοντέλο γίνεται fit στα δεδομένα που σπάσαμε προηγουμένως ανάλογα train/val και έτσι εξάγει εν τέλη το history.

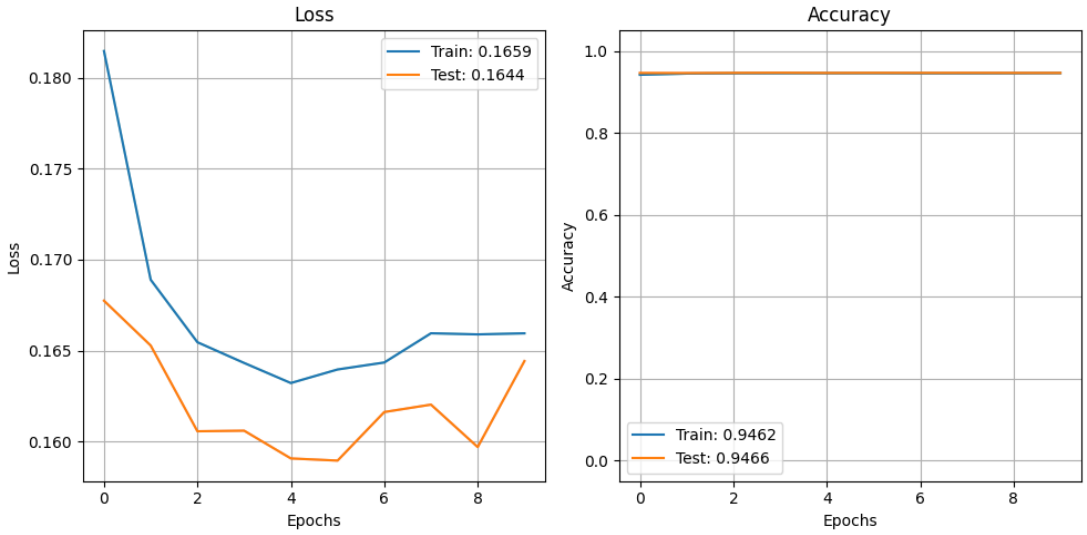
Το μοντέλο αργότερα γίνεται evaluate ώστε να έχουμε ένα loss-accuracy και εξάγει το plot του history.

```
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(MAX_NB_WORDS,100,input_length=X.shape[1]),
    tf.keras.layers.SpatialDropout1D(0.3),
   tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(100,dropout=0.2)),
    tf.keras.layers.Dense(2)
])

#model compile using RMSProp - learning rate 0.01 - 0.001
model.compile(loss=tf.keras.losses.BinaryCrossentropy(from_logits=True),optimizer=tf.keras.optimizers.RMSprop(learning_rate=0.001),
metrics=['accuracy'])
history = model.fit(X_train,Y_train, epochs=10, batch_size= 100, validation_data=(X_test, Y_test),  verbose=1).history
```

```
(812193, 250) (812193, 2)
2020-07-06 02:23:56.750855: W tensorflow/core/framework/cpu_allocator_impl.cc:81] Allocation of 812193000 exceeds 10% of free system memory.
Epoch 1/10
8122/8122 [==============================] - 2736s 337ms/step - loss: 0.1815 - accuracy: 0.9420 - val_loss: 0.1677 - val_accuracy: 0.9465
Epoch 2/10
8122/8122 [==============================] - 2713s 334ms/step - loss: 0.1689 - accuracy: 0.9449 - val_loss: 0.1653 - val_accuracy: 0.9463
Epoch 3/10
8122/8122 [==============================] - 2686s 331ms/step - loss: 0.1655 - accuracy: 0.9457 - val_loss: 0.1606 - val_accuracy: 0.9466
Epoch 4/10
8122/8122 [==============================] - 2690s 331ms/step - loss: 0.1643 - accuracy: 0.9458 - val_loss: 0.1606 - val_accuracy: 0.9470
Epoch 5/10
8122/8122 [==============================] - 2687s 331ms/step - loss: 0.1632 - accuracy: 0.9459 - val_loss: 0.1591 - val_accuracy: 0.9465
Epoch 6/10
8122/8122 [==============================] - 2693s 332ms/step - loss: 0.1640 - accuracy: 0.9459 - val_loss: 0.1590 - val_accuracy: 0.9468
Epoch 7/10
8122/8122 [==============================] - 2705s 333ms/step - loss: 0.1643 - accuracy: 0.9456 - val_loss: 0.1616 - val_accuracy: 0.9466
Epoch 8/10
8122/8122 [==============================] - 2725s 336ms/step - loss: 0.1660 - accuracy: 0.9456 - val_loss: 0.1620 - val_accuracy: 0.9467
Epoch 9/10
8122/8122 [==============================] - 2710s 334ms/step - loss: 0.1659 - accuracy: 0.9458 - val_loss: 0.1597 - val_accuracy: 0.9466
Epoch 10/10
8122/8122 [==============================] - 2706s 333ms/step - loss: 0.1659 - accuracy: 0.9462 - val_loss: 0.1644 - val_accuracy: 0.9466
2020-07-06 09:54:52.580932: W tensorflow/core/framework/cpu_allocator_impl.cc:81] Allocation of 812193000 exceeds 10% of free system memory.
Training Accuracy: 0.9471
```

Εικόνα 2 - 20 Epochs



Σχήμα 3 Loss,Accuracy

PREDICT ΓΙΑ ΤΟ TEST ΚΑΙ PREDICT ΓΙΑ RANDOM COMMENTS ΑΠΟ ΤΟ TEST.CSV ΤΟΥ KAGGLE

Χρησιμοποιώντας σε αυτό το σημείο το test που πήραμε από το σπάσιμο του αρχικού dataset, κάναμε τις προβλέψεις μας, και ήμασταν σε θέση να συγκρίνουμε τις πραγματικές τιμές της στήλης target με τις τιμές τις οποίες πρόβλεψε το μοντέλο μας.

Παρατηρήσαμε ότι το μοντέλο μας είχε επιτυχημένες προβλέψεις τις περισσότερες, και αυτό επιβεβαιώνει και το μεγάλο accuracy που υπολογίστηκε.

Για την σύγκριση των πραγματικών τιμών με τις τιμές που προέβλεψε το μοντέλο, φτιάξαμε μια νέα στήλη 'predicted' στο dataframe μας και τα αποθηκεύσαμε όλα σε ένα csv αρχείο για ευκολία.

Για επιπλέον έλεγχο των αποτελεσμάτων του μοντέλου μας αλλά και για τα ζητούμενα της εργασίας, επιλέξαμε κάποια σχόλια από το αρχείο test.csv του kaggle, και βάλαμε το μοντέλο να κάνει προβλέψεις. Χρειάστηκε να τροποποιηθούν και να επεξεργαστούν τα συγκεκριμένα comments ώστε να έχουν ακριβώς την ίδια μορφή που είχαν και τα δεδομένα με τα οποία εκπαιδεύτηκε το μοντέλο μας, έτσι χρειάστηκε για γίνουν τα βήματα που αναλύθηκαν στο (Προετοιμασία δεδομένων - Data Preprocessing). Τα σχόλια τα αποθηκεύσαμε σε μια λίστα η οποία αργότερα χρειάστηκε να μετατραπεί σε dataframe ώστε να ακολουθηθεί ακριβώς ίδια διαδικασία επεξεργασίας. Τα αποτελέσματα αυτά τα γράψαμε σε ένα αρχείο randompredictions.csv στην στήλη predict.

```
comments= ['What are the leftist snowflakes in Canada doing to stop
fat little Kim? Nothing.',
'I really hope like heck, there isnt a fucking Pokemon involved.',
'What a stupid fucking commenting system.','Is one of them gay?',
'Mean, spiteful, liars, power abusers, willing to pervert justice
for political power---all those traits Nixon is accused of having
are now resident in the Democrat Party and leftwing journalists.
Carl, YOU are Nixon. YOU are despicable.',
'Its a black mark on the previous administrations who allowed
massive illegal immigration to go on for far too long. Put the blame
where it should be.',
'There are no words for these senseless acts.. God help us all if
we do not learn that evil lurks even in the face of these teenagers.
Such violence against their peers, should make everyone take pause.
Be safe children, be safe. God says you can be the light of the
world.']
dftest   =   pd.DataFrame(dict(comment_text=comments))   #metatropi
listas se Dataframe
```

```
newLi = dftest['comment_text'].str.lower() #preprocess data gia na
ginei akrivos to keimeno opws auto pou egine to training
newLi  = newLi.apply(preprocessing_train)
newLi= newLi.str.replace('\d+', '')
newLi = pad_text(newLi, tokenizer)
dftest['predict'] = model.predict(newLi)[:, 1] #predict
dftest.to_csv('randompredictions.csv')
```

Τα αποτελέσματα από το αρχείο randompredictions.csv'φαίνονται στην εικόνα 3.

	comment_text	predict
2	What are the leftist snowflakes in Canada doing to stop fat little Kim? Nothing.	-2.0292866
3	I really hope like heck, there isnt a fucking Pokemon involved.	1.9682319
4	What a stupid fucking commenting system.	5.421771
5	Is one of them gay?	-1.8118893
6	Mean, spiteful, liars, power abusers, willing to pervert justice for political power--all those traits Nixon is accused of having are now resident in the Democrat Party and leftwing journalists. Carl, YOU a	-2.2481027
7	Its a black mark on the previous administrations who allowed massive illegal immigration to go on for far too long. Put the blame where it should be.	-3.5904748
8	There are no words for these senseless acts.. God help us all if we do not learn that evil lurks even in the face of these teenagers. Such violence against their peers, should make everyone take pausi	-3.278707

Στην στήλη predict τα αποτελέσματα τα οποία είναι >0 θεωρούμε ότι είναι TRUE άρα τοξικά ενώ όταν είναι <0 τότε FALSE μη τοξικά σχόλια.

- Βλέπουμε για παράδειγμα ότι το 'I really hope like heck, there isnt a fucking Pokemon involved.'ενώ ένας άνθρωπος ίσως δεν θα το θεωρούσε τόσο τοξικό σχόλιο, η μηχανή λόγο των εκφράσεων που χρησιμοποιεί το θεωρεί τοξικό.
- Ενώ το 10 σχόλιο 'What are the leftist snowflakes in Canada doing to stop fat little Kim? Nothing.' που κάποιος μπορεί να χαρακτήριζε σαν τοξικό σχόλιο λόγο του 'fat kim' το μοντέλο το θεωρεί σαν μη τοξικό.

AUC - BIAS

Σε αυτό το σημείο χρησιμοποιήθηκε το test dataframe, το μισό από το αρχικό μεγάλο αρχείο , όπως αναφέρθηκε και προηγουμένως, ώστε να μπορέσουμε να δούμε την σχέση του κάθε identity με το τελικό αποτέλεσμα. Δηλαδή το πόση επίδραση έχει στην τελική απόφαση του μοντέλου το κάθε identity, δεδομένου ότι το μοντέλο μας έχει ήδη εκπαιδευτεί σε κάποια σχόλια.

Αρχικά υπολογίζουμε το συνολικό ROC-AUC, και χρησιμοποιώντας αυτή την μετρική, μπορούμε να υπολογίσουμε τα biases του μοντέλου μας στις διαφορετικές identity ομάδες που έχουμε. π.χ white,black,men,women κλπ . Χρησιμοποιήσαμε τα identities που είχαν τουλάχιστον 5 true τιμές για πιο accurate αποτελέσματα.

Χρησιμοποιήθηκαν 3 διαφορετικές μετρικές bias για τις δοκιμές :

- SUBGROUP AUC
- POSITIVE SUBGROUP NEGATIVE AUC
- NEGATIVE SUBGROUP POSITIVE AUC

Μετά τον υπολογισμό των AUC χρησιμοποιήσαμε ένα heatmap για την οπτικοποίηση και την πιο εύκολη κατανόηση των αποτελεσμάτων μας.

Στο σχήμα 4 φαίνονται οι 3 διαφορετικές τιμές bias για το κάθε identity το οποίο έχει περισσότερα από 5 trues.

Βλέπουμε ότι κάποιες ομάδες, έχουν πιο χαμηλό AUC, όπως για παράδειγμα οι 'threat','bisexual','sexual_explicit'. Αυτό συμβαίνει γιατί το Negative Cross AUC είναι μικρότερο απο το Positive Cross AUC για τις συγκεκριμένες ομάδες. Έτσι φαίνεται ότι αυτές οι ομάδες έχουν περισσότερα false positives.

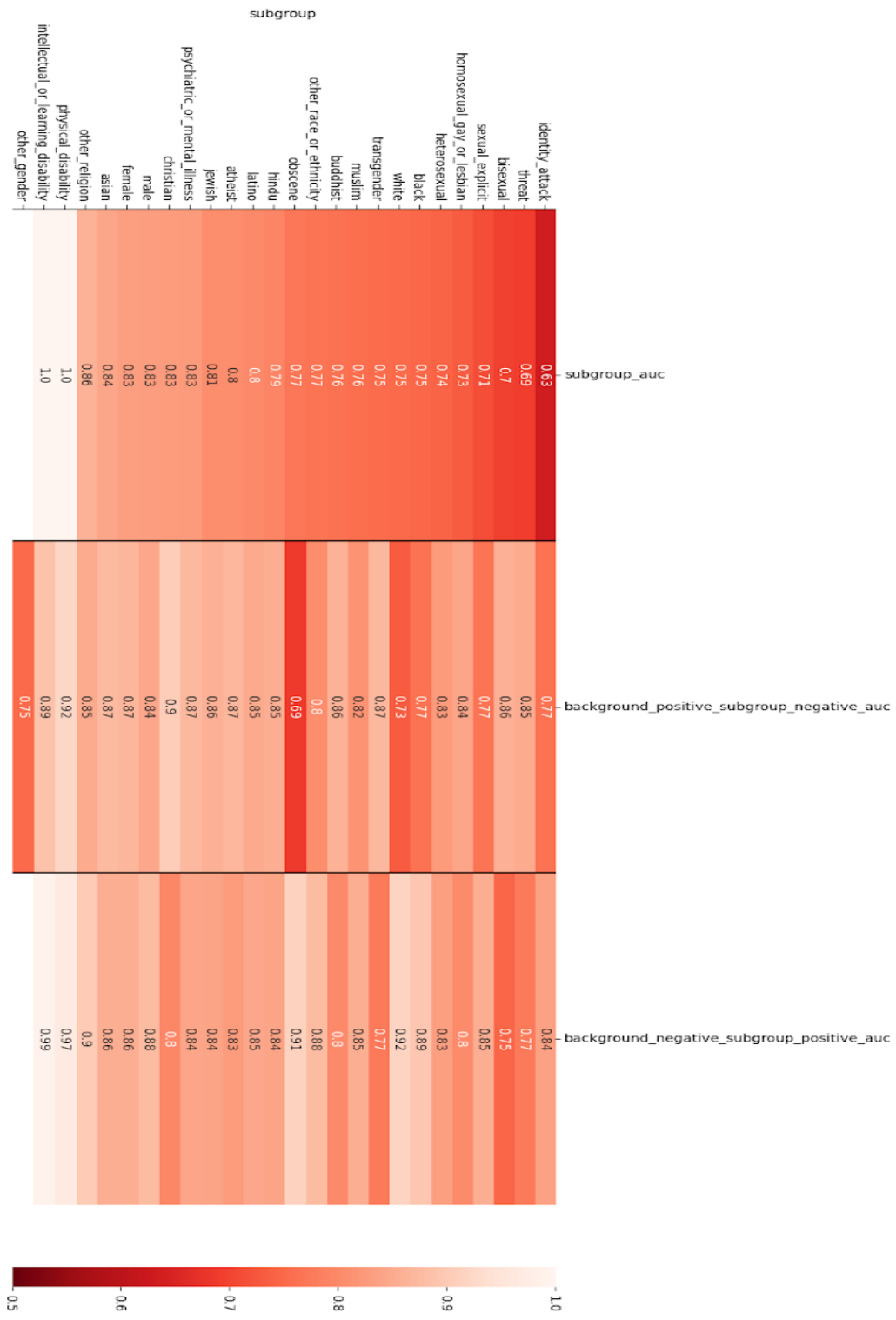
Έχουν δηλαδή έχουν περισσότερα non-toxic σχόλια που έχουν ψηλά ποσοστά toxicity απο τα σχόλια που αναφέρονται σε άλλες ομάδες.

Συμπερασματικά αν πάμε στο αρχείο μας, και βρούμε την στήλη π.χ 'bisexual' αν αυτή η τιμή είναι true, τότε έχει υψηλό ποσοστό και το τελικό target που προέβλεψε το μοντέλο μας να είναι true.

```
How many times identity_attack is true
False    895152
True       7285
Name: identity_attack, dtype: int64
How Many times target = true and identity_attack = true
False    895690
True       6747
Name: Bias_1, dtype: int64
Times target & identity_attack are both true and the prediction is Correct
False    901478
True        959
Name: Bias, dtype: int64
```

Από μία μικρή ανάλυση που κάναμε για την ομάδα identity_attack όντως απο τις 7285 φορές που είναι true, η στήλη target είναι true τις 6747 ένα αρκετά μεγάλο ποσοστό.

Παρόλα αυτά το predict απο τις 6747 φορές που θα έπρεπε να ήταν true, προβλέπει σωστά μόνο τις 959 και αυτό οφείλεται στο θόρυβο που δημιουργείται από τις υπόλοιπες ομάδες.



Σχήμα 4: AUC Heatmap for identities > 5 trues