Εργασία Εξαμήνου: «Ανάλυση και Διαχείριση Μεγάλων και Πολυδιάστατων Δεδομένων 2023-2024»

Ιωάννης Τσάμπρας ΣΜΗΝ 1066584

Τμήματα

- Προεπεξεργασία Clinical
  - Αριθμιτικοποίηση και Καθαρισμός
  - Συμπλήρωση Κενών
- Κατηγοριοποίηση Clinical
- Προεπεξεργασία Beacons
  - Ονοματοδοσία
  - Εξαγωγή Χαρακτηριστηκών
- Συσταδοποίηση Τελικού Dataset
  - Συσταδοποίηση
  - PCA

Προεπεξεργασία Clinical

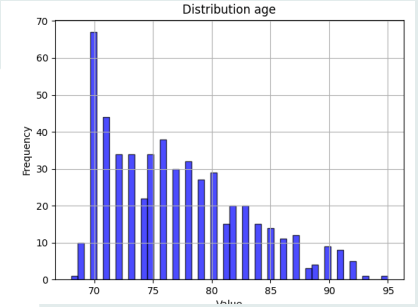
Αριθμιτικοποίηση και Καθαρισμός

Is this error?

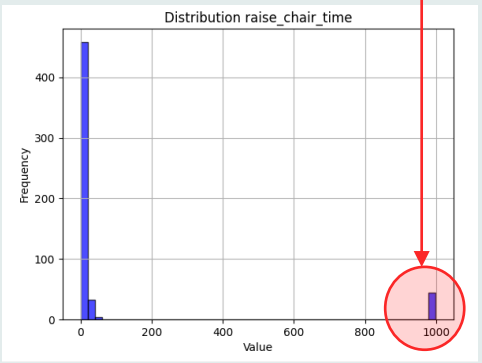
Μη αριθμητκές Στήλες

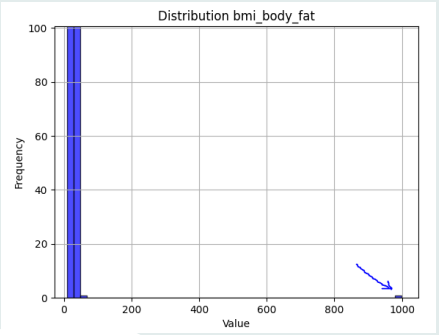
```
nom_dicts=[ #List indexes correspond to columns
    {"Frail":2,"Non frail":0,"Pre-frail":1},
    {"M":-1,"F":+1},
    {"No":0,"Yes":1},
    {'Sees well':0, 'Sees moderately':1, 'Sees poorly':2},
    {'Hears poorly':2, 'Hears well':0, 'Hears moderately':1},
    {"No":0,"Yes":1},
    {'<5 sec':0,'>5 sec':1},
    {'FALSE':0, 'TRUE':1},
    {"No":0,"Yes":1},
    {"No":0,"Yes":1},
    {"No":0,"Yes":1},
    {"No":0,"Yes":1},
    {'Permanent sleep problem':2, 'Occasional sleep problem':1, 'No sleep problem':0},
    {"No":0,"Yes":1},
    {"No":0,"Yes":1},
    {"No":0,"Yes":1},
    {"No":0,"Yes":1},
    {'2 - Bad':1, '4 - Good':3, '5 - Excellent':4, '3 - Medium':2, '1 - Very bad':0},
    {'2 - A little worse':1, '3 - About the same':2, '4 - A little better':3, '5 - A lot better':4,
    {'No':0, '< 2 h per week':1, '> 5 h per week':3, '> 2 h and < 5 h per week':2},
    {'Never smoked':0, 'Current smoker':2, 'Past smoker (stopped at least 6 months)':1},
]
```

Όσα κελιά δεν συνάυδουν με καμία τιμή του λεξικού μετατρέπονται σε κενά



Αριθμητικές Στήλες





Σύνθετη συμπεριφορά λεξικού με κανόνες

```
# from visualy inspecting the distributions plotted we establish the following thresholds and comments,
# the added "x" in front indicates that we will be capping the values because although outliers they do
thresholds=[
    None,
    None,
    None, ###but there are propably real but extreme outliers, maybe need to add log scale
    "x<15", ### maybe 999 represents living in hospital, 2 entries
    None,
    "x<55", ###maybe the 999 represents inability to lift, 40+ entries are not mistakes
    "x<65", ###maybe the 999 represents inability to get up, ~9 entries
    "x<50",  ###maybe the 999 represents inability to run, ~6 entries
    "x<7",  ###the 999 is propably just a mistake or inability to move in general, ~3 entries
    "<5",  ###the 999 is propably just wrong/ clearly eronius
    "<45",  ###the ~900 entry is physically impossible so clearly eronius
    "<65", ###the 999 entry is clearly eronius
    None,   ###there is one outlier but it is propably real
    ">10",   ###the ~ -400 entry is clearly eronius
    None,
    None,
    None,
    None,
    None,
    None,
    "<35",  ###don't know how to interpet those, propably eronius but 4 entries
    "<200", ###don't know how to interpet those, propably eronius 2 entries
    None,
    "x<200", ###maybe not eronius , it may signify something 10+ entries. there are also outliers
    "x<200", ###maybe not eronius , it may signify something 10+ entries. there are also outliers
    None,
    None,
```

Προεπεξεργασία Clinical
Συμπλήρωση Κενών

```
def fix_rows(rows):
    fixed_rows=[]
    from collections import defaultdict

    models = defaultdict(lambda: None)

    for i,row in enumerate(rows):
        print("Fixing row ",i)
        none_indices = [index for index, value in enumerate(row) if value is None] #get indexes where cell is empty
        semi_fixed_row=row.copy()
        for index_i in none_indices: #we fill empty values with avg
            semi_fixed_row[index_i]=avg_values[index_i]

        for index_i in none_indices:
            prediction_column_index=index_i
            if models[index_i]==None:
                print(f"model for column {index_i} is being created")
                trained_model, scaler = train_neural_network(full_entries, target_column_index=prediction_column_index)
                models[index_i]=[trained_model,scaler]
            else:
                print(f"model for column {index_i} is cached")



            # Now, you can use the trained model and scaler to make predictions:
            trained_model,scaler=models[index_i]
            predicted_value = predict_cell_value(trained_model, scaler, semi_fixed_row, target_column_index=prediction_column_index)
            semi_fixed_row[index_i]=predicted_value
        fixed_row=semi_fixed_row.copy()
        fixed_rows.append(fixed_row)
            #print(f'Predicted value: {predicted_value}, Ground Truth: {my_row[prediction_column_index]}')
    return fixed_rows
fixed=fix_rows(problematic_entries)
```

Naive method: Replace Empty Cells with average of column

New method: Create neural network based predictors for each column and replace values with predictions based on the rest of the row

```
def train_neural_network(dataset, target_column_index, epochs=2000, batch_size=64):
    # Extract features and target variable
    features = np.array([row[:target_column_index] + row[target_column_index+1:] for row in dataset])
    target = np.array([row[target_column_index] for row in dataset])

    # Split the dataset into training and testing sets
    features_train, features_test, target_train, target_test = train_test_split(features, target, test_size=0.2, random_state=42)

    # Standardize the features
    scaler = StandardScaler()
    features_train_scaled = scaler.fit_transform(features_train)
    features_test_scaled = scaler.transform(features_test)

    # Build a simple neural network model
    model = tf.keras.Sequential([
        tf.keras.layers.Dense(8192, activation='relu', input_shape=(features_train_scaled.shape[1],)),
        tf.keras.layers.Dense(1)
    ])

    # Compile the model
    model.compile(optimizer='adam', loss='mean_squared_error')

    # Train the model
    model.fit(features_train_scaled, target_train, epochs=epochs, batch_size=batch_size, validation_split=0.2, verbose=0)

    # Evaluate the model on the test set
    loss = model.evaluate(features_test_scaled, target_test, verbose=0)
    print(f'Test Loss: {loss}')

    return model, scaler
```

In this work:

- We trained the predictors only on the full columns (~half the dataset)
- We cached each model for reuse
- Rounded outputs for previously non-numerical columns in the nearest integer
- Each predictor is identical to the rest in terms of hyperparameters
- We also applied the naive method for comparison

Κατηγοριοποίηση Clinical

Αξιοποιήθηκε KNN και Νευρωνικό δίκτυο

Για τον **KNN** τα αποτελέσμτα δεν αλλάζουν, με βάση το dataset, με μέγιστο accuracy **0.66** και μέσο **0.59** για τους πρώτους **40** γείτονες.

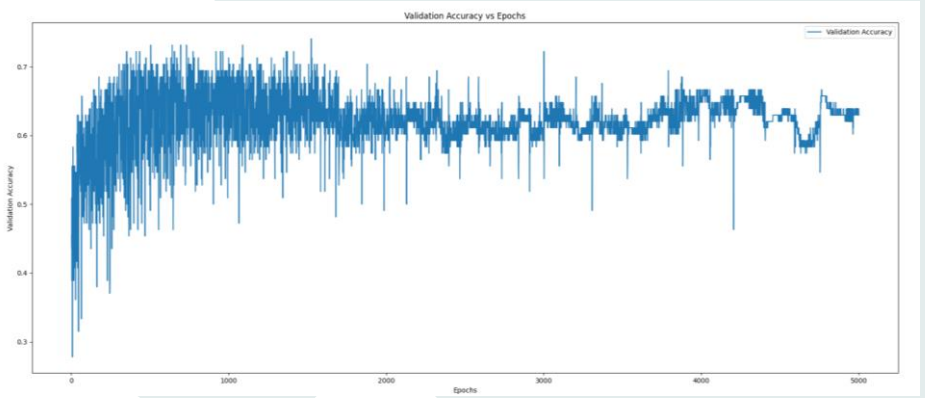
Για το **νευρωνικό δίκτυο** πραγματοποιήθηκαν αρχικά **8** και ύστερα επιπλέον **16** trainings με κάθε dataset και από αυτά προέκυψαν οι παρακάτω **average validation accuracies**:

```
#in the 8 runs
#with predictor dataset 0.7337962910532951 [0.7
#with simple averages   0.6296296268701553 [0.6

#in the 16 runs
#with predictor dataset 0.7303240746259689 [0.7
#with simple averages   0.6238425895571709 [0.5
```

Αύξηση ακρίβειας κατά **11%** με τη χρήση του "predicted dataset"

Για τους παράμετρους του νευρωνικού έγιναν πολλαπλές δοκιμές και κατέληξα σε :

- 3 layers (1024,128,3)
- Learning rate 0.0001
- 2000 epochs
- Batchsize 64

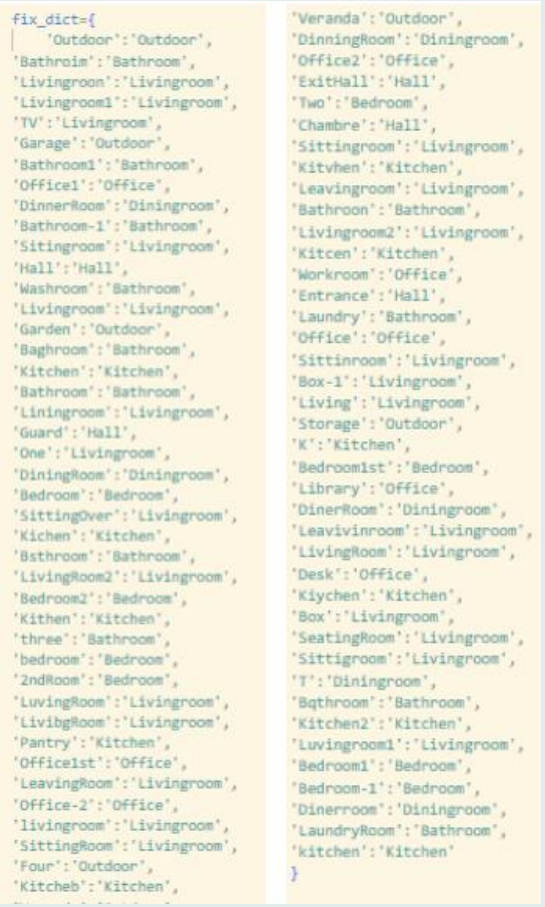


Προεπεξεργασία Beacons

Ονοματοδοσία

Αρχικά καθαρίζουμε τις εγγραφές από μη προβλεπόμενα IDs και ύστερα συλλέγουμε όλες τις διαφορετικές ονομασίες δωματίων.

Ύστερα χειροκίνητα κατασκευάζουμε ένα λεξικό αντιστοίχησης όλων των διαφορετικών ονομασιών με τις πραγματικές ονομασίες που επιλέξαμε οι οποίες είναι 8:



• Bathroom • Bedroom • Diningroom • Hall • Kitchen • Livingroom • Office • Outdoor

Επιπλεόν συμπτύσουμε τις στήλες ημερομηνίας και ώρας και τις μετατρέπουμε σε UNIX time για να διευκολύνουμε την επεξεργασία τους αργότερα.

Προεπεξεργασία Beacons

Εξαγωγή Χαρακτηριστηκών

Με σειρά βημάτων υπολογίζουμε το ποσοστό διαμονής του κάθε συμμετέχοντα στα επιμέρους δωμάτια επί του συνολικού.

Προσέχουμε την αλλαγή της ημέρας και βάζουμ όριο τα μεάνυχτα.

```
1   part_id,Bathroom,Bedroom,Diningroom,Hall,Kitchen,Livingroom,Office,Outdoor
214 3061,10.9,11.7,25.3,0.0,4.4,11.7,0.0,35.7
215 3062,0.0,0.0,21.7,26.9,24.0,14.3,0.0,12.9
216 3064,2.1,25.0,0.0,0.0,18.9,53.7,0.0,0.0
217 3065,0.7,18.3,44.5,0.0,7.0,9.4,0.0,19.7
218 3066,5.3,7.0,0.0,0.0,7.7,59.8,0.6,19.2
219 3067,0.0,0.0,2.2,61.3,27.5,0.1,0.3,8.2
220 3068,0.0,18.2,12.2,0.0,35.2,11.2,19.9,3.0
221 3070,3.6,0.0,32.5,0.0,29.2,10.0,0.5,23.9
```

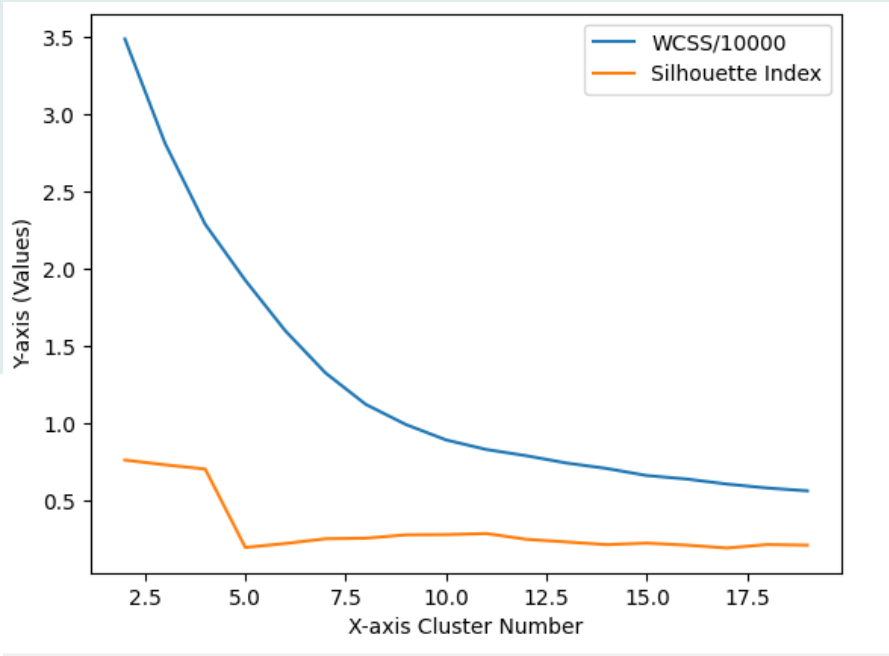
```python
#func to check if it's a new day
def has_day_changed(timestamp1, timestamp2):  #
    # Convert Unix timestamps to datetime objects
    date1 = datetime.utcfromtimestamp(timestamp1).date()
    date2 = datetime.utcfromtimestamp(timestamp2).date()

    # Compare the day components
    return date1 != date2


def seconds_until_midnight(unix_timestamp):
    # Convert Unix timestamp to datetime object
    dt = datetime.utcfromtimestamp(unix_timestamp)

    # Calculate midnight of the next day
    midnight = datetime(dt.year, dt.month, dt.day) + timedelta(days=1)

    # Calculate the time difference in seconds
    seconds_remaining = (midnight - dt).total_seconds()

    return int(seconds_remaining)

def seconds_from_start_of_day(unix_timestamp):
    # Convert Unix timestamp to datetime object
    dt = datetime.utcfromtimestamp(unix_timestamp)

    # Calculate midnight of the same day
    start_of_day = datetime(dt.year, dt.month, dt.day)

    # Calculate the time difference in seconds
    seconds_elapsed = (dt - start_of_day).total_seconds()

    return int(seconds_elapsed)
```

Συσταδοποίηση Τελικού Dataset

Τα δεδομένα μας αρχικά διαιρούνται κατά στήλες με τον μέσο της στήλης για να **εξισώσουμε τη συνεισφορά κάθε στήλης** στην γεωμτερική απεικόνιση των εγγραφών.

Από τα δεδομένα μας αφαιρούμε τις στήλες των IDs και το Fried-Index.

Για την συσταδοποίηση των δεδομένων αξιοποιούμε **k-means** και δοκιμάζουμε πολλαπλά πλήθη κεντροειδών. Αξιοποιούμε σε κάθε πιθανό πλήθος την μετρική της **«αδράνειας» και το sil index** και τα τοποθετούμε σε γραφική παράσταση για να βρούμε τον αγκώνα του γραφήματος και το σχετικό sil index.



Οπτικά εντοπίζουμε τον αγκώνα του γραφήματος στα **9 κεντροειδή** και το αντίστοιχο **sil index είναι 0.291**, σχετικά ασθενές.

PCA

Επιπλέον παράγουμε ένα δεύτερο set δεδομένων με χρήση PCA για να **μικρύνουμε τη διαστατικότητα**. Για το πλήθως των components **"k"** δοκιμάστηκαν διάφορες επιλογές, όπως φαίνεται στον παρακάτω πίνακα, και συναντάμε τα 9 κεντροειδή στα 8 components (όσα περισσότερα components και αν προσθέσουμε τα κεντροειδή θα παραμείνουν 9 αφού τόσα βρήκαμε και πριν το dimensionality reduction).

```
#by experiementing with pca we gain the following:
#components  #elbow  #silhouette value
#2           #2      0.965
#3           #4      0.775
#4           #5      0.778
#5           #6      0.771
#6           #6      0.771
#7           #7      0.697
#8           #9      0.673
#9           #9      0.641
#10          #9      0.564
#11          #9      0.510
#12          #9      0.472
```