Osteoporotic fracture prediction and risk variable selection

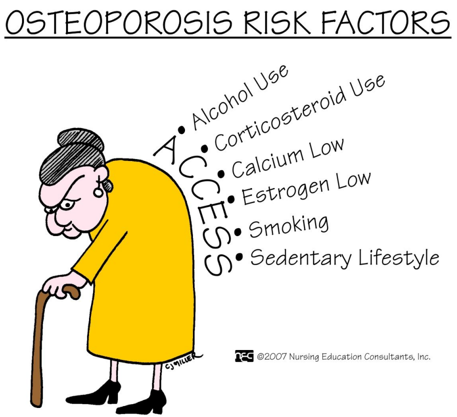
O Mercy Akinloye & Denise Bock

January. 08. 2024

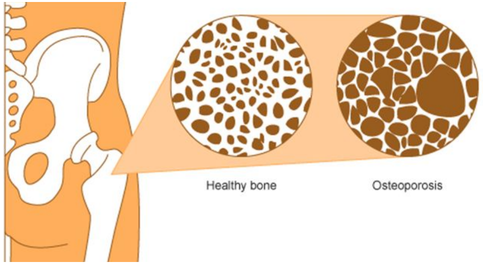
OSTEOPOROSIS

“Porous bone condition” - most common chronic and metabolic bone disease

Occurs as a result of increased osteoclasts activity and decreased osteoblasts activity which results in increased bone breakdown and decreased bone formation



Characterized by:
- Low bone density or mass
- Deterioration of bone tissue
- Increased susceptibility to fractures



People at risk:
- Risk (↑↑↑) with age
- Females > Males

Fractures

Backache

Low trauma fractures

SOF

SOF: Study of Osteoporotic Fracture

- American, multicenter, prospective study
- 10, 366 women, 65 years or older
    - clinical visitation every 2 years; ~ 20 years
    - information on:
        - *bone mineral density, cognitive exams and more...*

Dataset: https://sofonline.ucsf.edu/

Task / Problem:

- Rank features (feature importance)
- Use these features to predict fracture risk
- ~~Trend in feature changes over time contribute to fracture risk~~

Literature Review

<u>Source 1:</u> Jin H, Lu Y, Harris ST, Black DM, Stone K, Hochberg MC, Genant HK (2004). **Classification algorithms for hip fracture prediction based on recursive partitioning methods.** *Med Decis Making 24*(4):386-98. doi: 10.1177/0272989X04267009. PMID: 15271277 - PubMed

- **<u>Objective:</u>** cost-saving classification approach using 43 predictive variables for 5-year hip fracture risk with equivalent diagnostic utility as the use of a robust optimum classification rule
- **<u>Methods:</u>** generation of a cost-saving classification rule and conduction of a validation study
- **<u>Outcomes:</u>** cost-saving classification rule is statistically non-inferior to the robust optimum classification rule
- **<u>Relation to the Project:</u>** hints for variable selection based on variable importance, theoretical background on clinically important variables and decision making

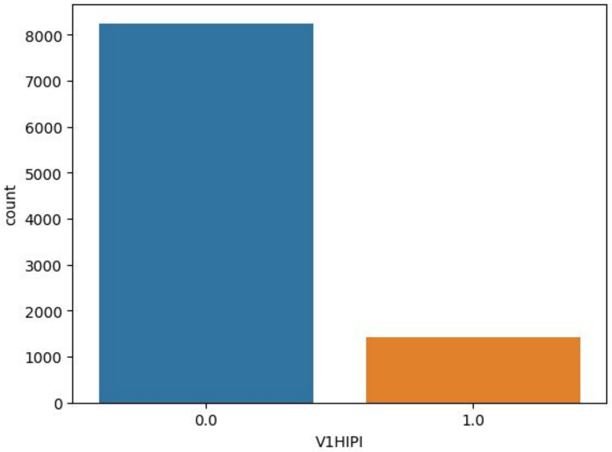
<u>Source 2</u>: Stroke Prediction Using Machine Learning - Kaggle

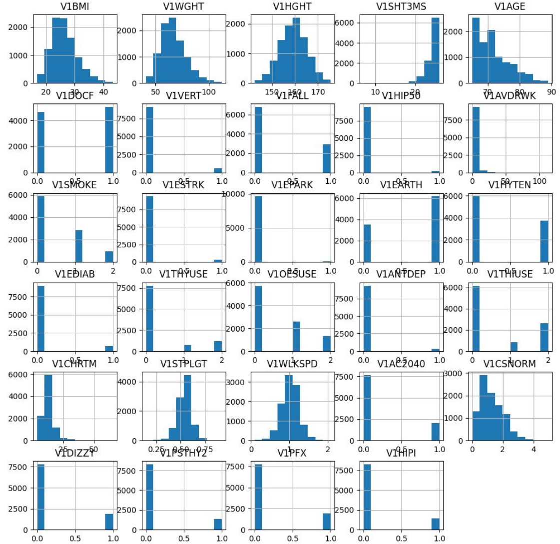
- **<u>Objective</u>:** Analyze health data for feature selection and stroke prediction
- **<u>Methods</u>:** Scikit-learn, Logistics Regression, SVM, RandomForestClassifier, XGBoost classifier, ROC/AUC score, confusion matrix
- **<u>Outcomes</u>:** Random Forest performed the best in terms of accuracy
- **<u>Relation to the Project</u>:** (National Health & Nutrition Examination Survey) dataset has a similar structure to SOF dataset - provides some method for balancing dataset which could be useful for our own imbalanced dataset - feature selection and prediction are also included in this analysis - structure is similar to our intended analysis structure

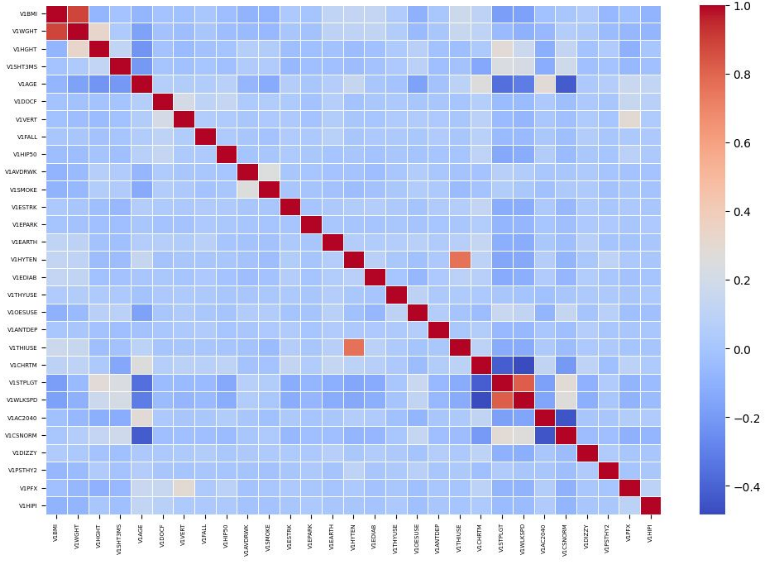
Dataset Characteristics

- Number of samples (participants): 9704 → 9666
- Number of features (variables): 32 → 29 → 26

- Removed features with >1000 missing values
- Dropped participants with missing HIPI information
- Eventually, removed weight and height
- Kept correlated but non redundant variables
- Median imputation for missing values



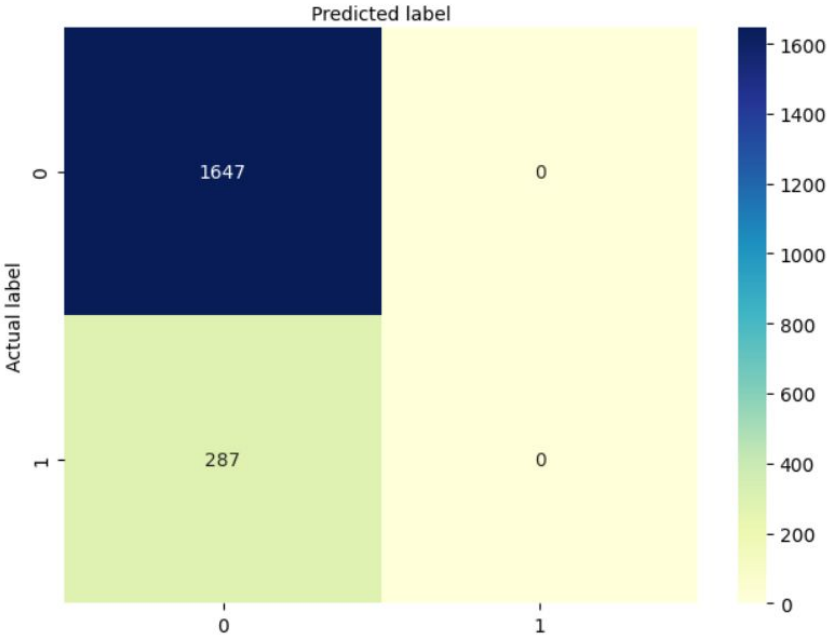




Baseline Model

```python
#baseline model
from sklearn.linear_model import LogisticRegression

# instantiate the model (using the default parameters)
logreg = LogisticRegression(random_state=16, max_iter = 5000)

# fit the model with data
logreg.fit(X_train, y_train)

y_pred = logreg.predict(X_test)

# Evaluate accuracy
base_accuracy = accuracy_score(y_test, y_pred)
print(f"BASELINE LR Accuracy: {base_accuracy}")
```

```
BASELINE LR Accuracy: 0.8521199586349535
```



Sensitivity : 0.003
Specificity : 1

- **Sensitivity** is the proportion of actual positive cases that are correctly identified by the model (true positives)
- **Specificity** is the proportion of actual negative cases that are correctly identified by the model (true negatives)

Undersampling

```
#Undersampling majority class
from imblearn.under_sampling import NearMiss
from sklearn.model_selection import train_test_split

# Initialize NearMiss
nm = NearMiss()

# Undersample the majority class
X_train_undersampled, y_train_undersampled = nm.fit_resample(X_train, y_train)
```

Original Hip Fracture Distribution:

0 6600

1 1132

Resampled Hip Fracture Distribution:

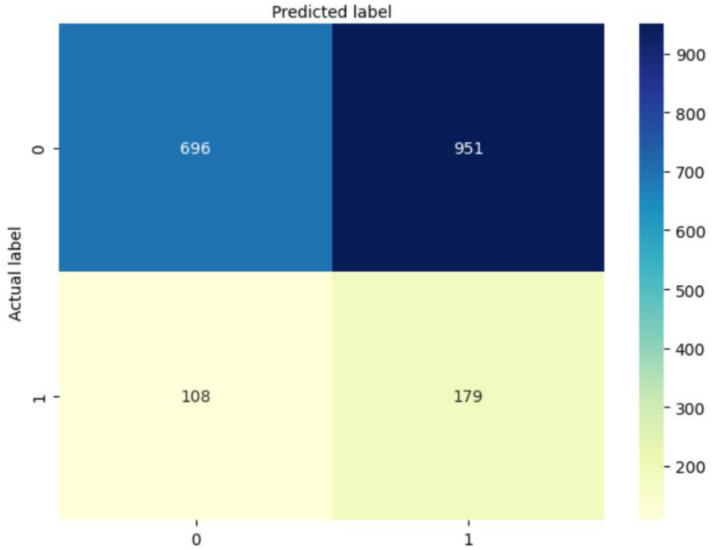
0 1132

1 1132

Baseline model - undersampled

```
#baseline model - all variables
from sklearn.linear_model import LogisticRegression

# instantiate the model (using the default parameters)
logreg_all = LogisticRegression(random_state=16, max_iter = 5000)

# fit the model with data
logreg_all.fit(X_train_undersampled, y_train_undersampled)

y_pred_all = logreg_all.predict(X_test)

# Evaluate accuracy
base_accuracy_all = accuracy_score(y_test, y_pred_all)
print(f"BASELINE LR Accuracy: {base_accuracy}")
```

```
BASELINE LR Accuracy: 0.5863495346432265
```



Sensitivity : 0.631
Specificity : 0.425

Model Improvement: Feature Selection with Random Forest

```python
# feature reduction
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier
import pandas as pd

# Create a Random Forest classifier
rf_classifier_und = RandomForestClassifier(n_estimators=100, random_state=42)

# Re-Train the classifier
rf_classifier_und.fit(X_train_undersampled, y_train_undersampled)

# Get feature importances
feature_importances = rf_classifier_und.feature_importances_

# Add feature im,[rtamc]
feature_importance_df = pd.DataFrame({'Feature': X_train_undersampled.columns, 'Importance': feature_importances})

# Sort the DataFrame by importance in descending order
feature_importance_df = feature_importance_df.sort_values(by='Importance', ascending=False)

# choose top 10 variables
top_10 = feature_importance_df[:10]
print(top_10)
```

	Feature	Importance
18	V1CHRTM	0.140339
7	V1AVDRWK	0.140311
0	V1BMI	0.119618
20	V1WLKSPD	0.092690
22	V1CSNORM	0.092397
2	V1AGE	0.091276
1	V1SHT3MS	0.069376
19	V1STPLGT	0.055929
25	V1PFX	0.023830
3	V1DOCF	0.017577

Model Improvement: Neural Network

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score

# Build a simple neural network for logistic regression
model = Sequential()
model.add(Dense(units=64, input_dim=10, activation='relu'))
model.add(Dense(units=32, activation='relu'))
model.add(Dense(units=1, activation='sigmoid'))

# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Train the model
model.fit(X_train_undersampled_top, y_train_undersampled, epochs=40, batch_size=1, verbose=1)
```

Model Comparison - LR, RF, NN (undersampled, top features)

Model	Accuracy	Sensitivity	Specificity
Logistic Regression	0.477	0.645	0.447
Random Forest	0.423	0.693	0.352
Neural Network	0.522	0.488	0.528

Challenges & Errors

1. Could not significantly improve the accuracy
2. Unstructured coding
3. Complex dataset
4. Mistook the good accuracy of the original baseline model for a good result

Future Work

1. Gauging what’s actually feasible with the data that you have: How important is feature reduction?
2. Other important variables that were assessed at other visitation period could be considered, e.g. bone mineral density
3. Other imputation methods might be more suitable to this dataset

THANK YOU FOR YOUR ATTENTION!

ANY QUESTIONS? COMMENTS?