PREDICTING RETIREMENT AGE

EDGAR ABASOV

INTRODUCTION

- Project focused on predicting retirement age using advanced machine learning techniques applied to the Survey of Health, Ageing and Retirement in Europe (SHARE) Wave 6 dataset

SHARE
SURVEY OF HEALTH, AGEING AND RETIREMENT IN EUROPE

PROJECT OBJECTIVES

- Develop accurate classification **models** to predict **retirement age** based on health, socio-economic, and policy-related factors.
- Identify key predictors of **retirement age** (e.g., health status, income, pension policies).
- Provide **insights** for policymakers on aging and retirement trends.
- **Research Question:** How do bio-medical and socio-economic factors interact to influence retirement age in European populations?

DATASET DESCRIPTION

- Source: SHARE Wave 6 (2014-2015), part of a 10-wave longitudinal study.
- Key Features:
- Demographic: Age, gender, education, marital status.
- Health: Physical/mental health, biomarkers
- Socio-economic: Income, wealth, employment status, pension details.
- Retirement: Self-reported retirement age, labor market exit reasons.
- Sample Size: ~68,000 individuals across 18 countries.

	ep064d9_7_w9	exrate_2	it001__10	ep064d1_7_w9	language	ph006d20_12	gs009__8	it003__10	br015__3	dn042__6	ep033__7	ep036__7	ep029__7	gs006__8	ac023__1	ep009__7	ac035d8_1
176	0.0	1.0679	1.0	0.0	20	0.0	45.0	4.0	2.0	1	2.0	5.0	4.0	48.0	1.0	3.0	1.0
251	0.0	1.0679	1.0	0.0	20	0.0	66.0	4.0	2.0	1	2.0	5.0	4.0	62.0	1.0	1.0	1.0
761	0.0	1.0000	1.0	1.0	17	0.0	33.0	4.0	1.0	1	3.0	1.0	3.0	30.0	3.0	1.0	0.0
755	0.0	1.0000	1.0	0.0	17	0.0	49.0	4.0	2.0	2	2.0	1.0	1.0	44.0	3.0	1.0	1.0
730	0.0	1.0000	5.0	0.0	17	1.0	22.0	4.0	2.0	2	1.0	1.0	3.0	21.0	3.0	1.0	1.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
118	0.0	1.0000	5.0	1.0	11	0.0	56.0	6.0	1.0	1	3.0	5.0	2.0	48.0	3.0	1.0	1.0
446	0.0	1.0000	1.0	0.0	12	0.0	45.0	4.0	2.0	1	1.0	1.0	4.0	45.0	2.0	1.0	0.0
1062	0.0	1.0000	1.0	1.0	16	1.0	28.0	6.0	4.0	2	2.0	1.0	3.0	25.0	2.0	2.0	1.0
353	1.0	1.0000	5.0	1.0	12	0.0	41.0	5.0	1.0	2	4.0	1.0	2.0	35.0	2.0	2.0	1.0
66	0.0	1.0000	5.0	1.0	11	0.0	45.0	6.0	1.0	1	3.0	1.0	3.0	39.0	1.0	1.0	1.0

METHODOLOGY- DATA PREPROCESSING

- Converting from the source format to a more effective format
- Selecting variables: with potential usability level
- Handled missing values using multiple imputation.
- Encoded categorical variables (e.g., one-hot encoding for education).
- Normalised continuous variables (e.g., income, health scores).

MODEL SELECTION

Classification Algorithms:

- Logistic Regression: Baseline for interpretability.
- Random Forest: Captures non-linear relationships.
- XGBoost: High performance for complex datasets.

Feature Selection:

- Experiment: sklearn tree view suggestion
- Final: SHAP

Tools: Python (pandas, scikit-learn, XGBoost)

```
    'XGBoost': {
        'model': xgb.XGBClassifier(random_state=42, eval_metric='logloss', use_label_encoder=False,
                                   objective='binary:logistic'),
        'params': {
            'classifier__n_estimators': [100, 300],
            'classifier__learning_rate': [0.05, 0.1],
            'classifier__max_depth': [3, 5],
            'classifier__subsample': [0.8],
            'classifier__colsample_bytree': [0.8],
            'classifier__gamma': [0],
            'classifier__reg_lambda': [1],
            'classifier__reg_alpha': [0]
        }
    },
    'LightGBM': {
        'model': lgb.LGBMClassifier(random_state=42, objective='binary'),
        'params': {
            'classifier__n_estimators': [100, 300],
            'classifier__learning_rate': [0.05, 0.1],
            'classifier__num_leaves': [20, 31],
            'classifier__max_depth': [-1],
            'classifier__min_child_samples': [20],
            'classifier__subsample': [0.8],
            'classifier__colsample_bytree': [0.8],
            'classifier__reg_alpha': [0],
            'classifier__reg_lambda': [0]
        }
    },
    'CatBoost': {
        'model': cb.CatBoostClassifier(random_state=42, verbose=0, eval_metric='Logloss'),
        'params': {
            'classifier__iterations': [100, 300],
            'classifier__learning_rate': [0.05, 0.1],
            'classifier__depth': [4, 6],
            'classifier__l2_leaf_reg': [3],
            'classifier__border_count': [32],
            'classifier__loss_function': ['Logloss']
        }
    },
    'MLP Classifier': {
        'model': MLPClassifier(random_state=42, max_iter=2000),
        'params': {
            'classifier__hidden_layer_sizes': [(50,), (100,)],
            'classifier__activation': ['relu'],
            'classifier__alpha': [0.0001],
            'classifier__learning_rate_init': [0.001],
            'classifier__solver': ['adam']
        }
    },
    'Baseline (Stratified Dummy)': {
```

TENSORFLOW CLASSIFICATION

```python
# Step 1: Train and save a base model (simulating a pretrained model)
base_model: Any = models.Sequential([
    layers.Input(shape=(X_train.shape[1],)),
    layers.Dense(256, activation='relu'),
    layers.BatchNormalization(),
    layers.Dropout(0.4),
    layers.Dense(128, activation='relu'),
    layers.BatchNormalization(),
    layers.Dense(1, activation='sigmoid')
])

base_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
base_model.fit(X_train, y_train, epochs=10, batch_size=32, validation_split=0.2)

# Save the base model
base_model.save('base_model.h5')

# Step 2: Load the pretrained model and add custom layers
pretrained_model: Any = tf.keras.models.load_model('base_model.h5')
pretrained_model.trainable = False  # Freeze pretrained layers

# Build new model with custom layers
inputs: Any = layers.Input(shape=(X_train.shape[1],))
x: Any = pretrained_model(inputs, training=False)  # Use pretrained model as a layer
x: Any = layers.Dense(64, activation='relu')(x)  # Custom layer
x: Any = layers.Dropout(0.3)(x)
x: Any = layers.Dense(32, activation='relu')(x)
outputs: Any = layers.Dense(1, activation='sigmoid')(x)  # Binary classification: 0 ("one") or 1 ("two"

new_model: Any = models.Model(inputs, outputs)

# Compile new model
new_model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001),
                  loss='binary_crossentropy', metrics=['accuracy'])

# Train new model (only custom layers)
new_model.fit(X_train, y_train, epochs=10, batch_size=32, validation_split=0.2)

# Optional: Fine-tune by unfreezing pretrained layers
pretrained_model.trainable = True
new_model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=1e-5),  # Lower learning rate
                  loss='binary_crossentropy', metrics=['accuracy'])
new_model.fit(X_train, y_train, epochs=5, batch_size=32, validation_split=0.2)

# Evaluate model
loss: Any, accuracy: Any = new_model.evaluate(X_test, y_test)
print(f"Test accuracy: {accuracy:.4f}")

# Example prediction with class names
sample: Any = X_test[0:1]
prediction: Any = new_model.predict(sample)
class_names: list[str] = ['Not thinking about Retiring',
               'Wants to Retire']  # Map 0 to "one", 1 to "two"
predicted_class: str = class_names[int(prediction[0][0] > 0.5)]
print(f"Prediction: {predicted_class}")
```

```
18/30
━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9541 - loss: 0.1213 - val_accuracy: 0.9438 - val_loss: 0.
19/30
━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9653 - loss: 0.0692 - val_accuracy: 0.9438 - val_loss: 0.
20/30
━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9742 - loss: 0.0701 - val_accuracy: 0.9375 - val_loss: 0.
21/30
━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9759 - loss: 0.0473 - val_accuracy: 0.9375 - val_loss: 0.
22/30
━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9570 - loss: 0.1105 - val_accuracy: 0.9125 - val_loss: 0.
23/30
━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9551 - loss: 0.0886 - val_accuracy: 0.9375 - val_loss: 0.
24/30
━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9686 - loss: 0.0901 - val_accuracy: 0.9187 - val_loss: 0.
25/30
━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9670 - loss: 0.0848 - val_accuracy: 0.9312 - val_loss: 0.
26/30
━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9726 - loss: 0.0590 - val_accuracy: 0.9375 - val_loss: 0.
27/30
━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9690 - loss: 0.0936 - val_accuracy: 0.9500 - val_loss: 0.
28/30
━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9808 - loss: 0.0588 - val_accuracy: 0.9438 - val_loss: 0.
29/30
━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9743 - loss: 0.0671 - val_accuracy: 0.9625 - val_loss: 0.
30/30
━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9548 - loss: 0.0998 - val_accuracy: 0.9312 - val_loss: 0.
━━━━━━━━━━━━━━━━━━━━ 1s 94ms/step - accuracy: 0.9107 - loss: 0.1484
ccuracy: 0.9150
━━━━━━━━━━━━━━━━━━━━ 0s 416ms/step
tion: Negative
```

TRAINING AND EVALUATION

Data Split:

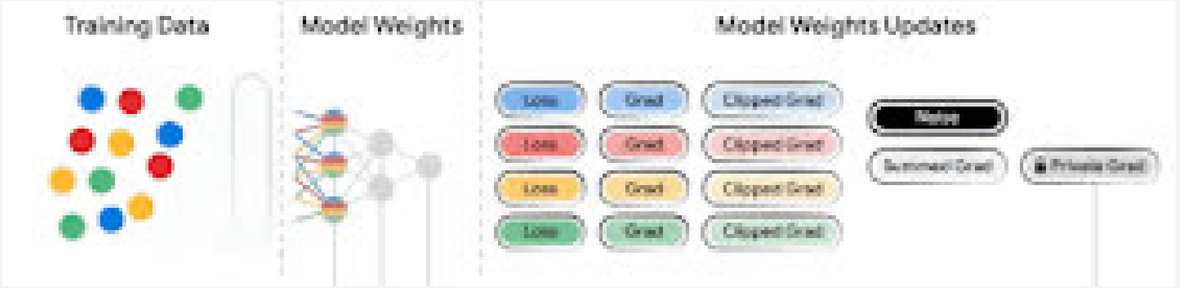
- 80% training, 20% test.

Evaluation Metrics:

- Accuracy, precision, recall, F1-score.
- AUC-ROC for class discrimination.

Techniques:

- 3-fold cross-validation for robustness.
- GridSearch
- Manual handling imbalanced classes of target
- Visualization: Confusion matrix and ROC curve (include sample charts), model comparision bars

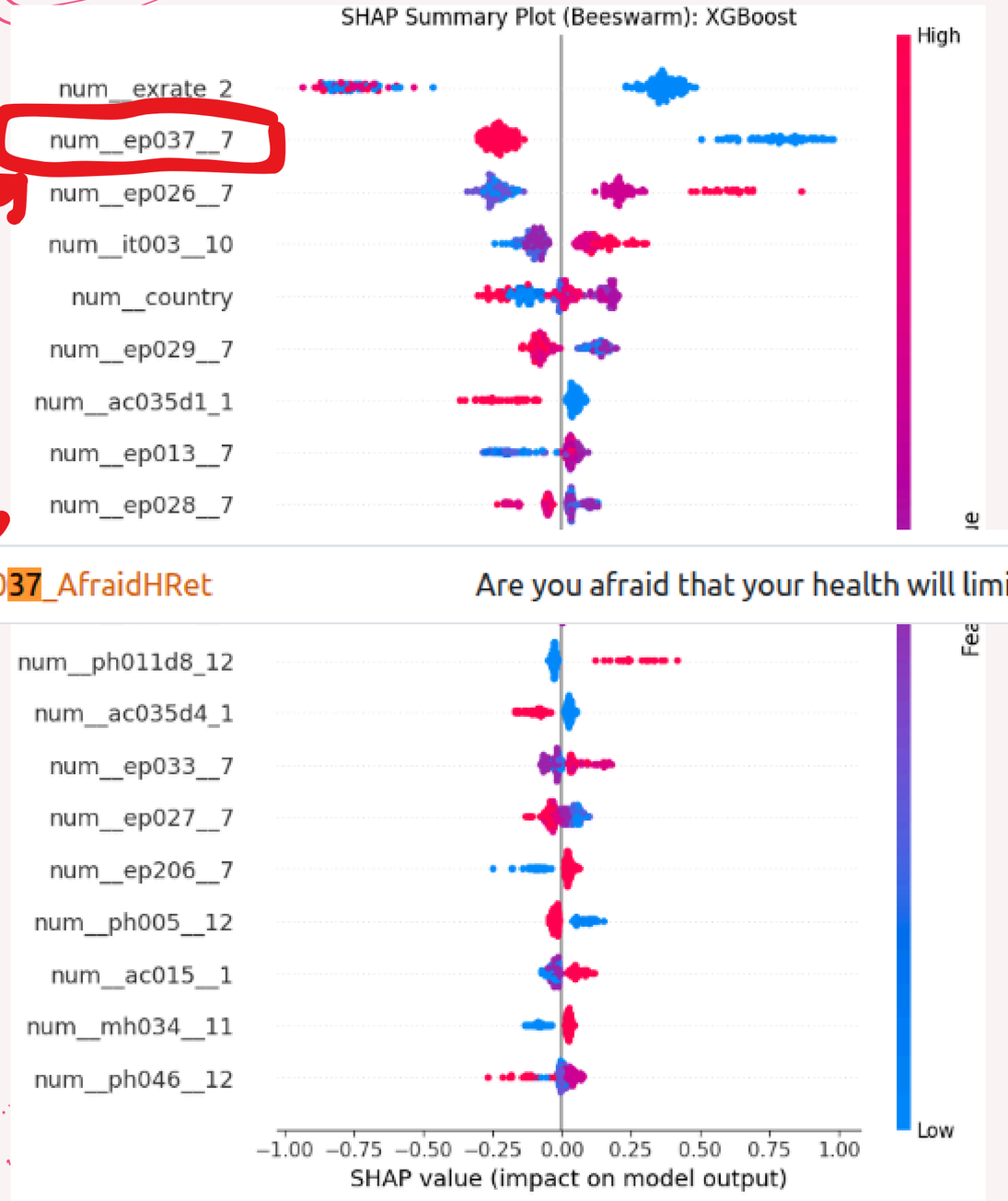


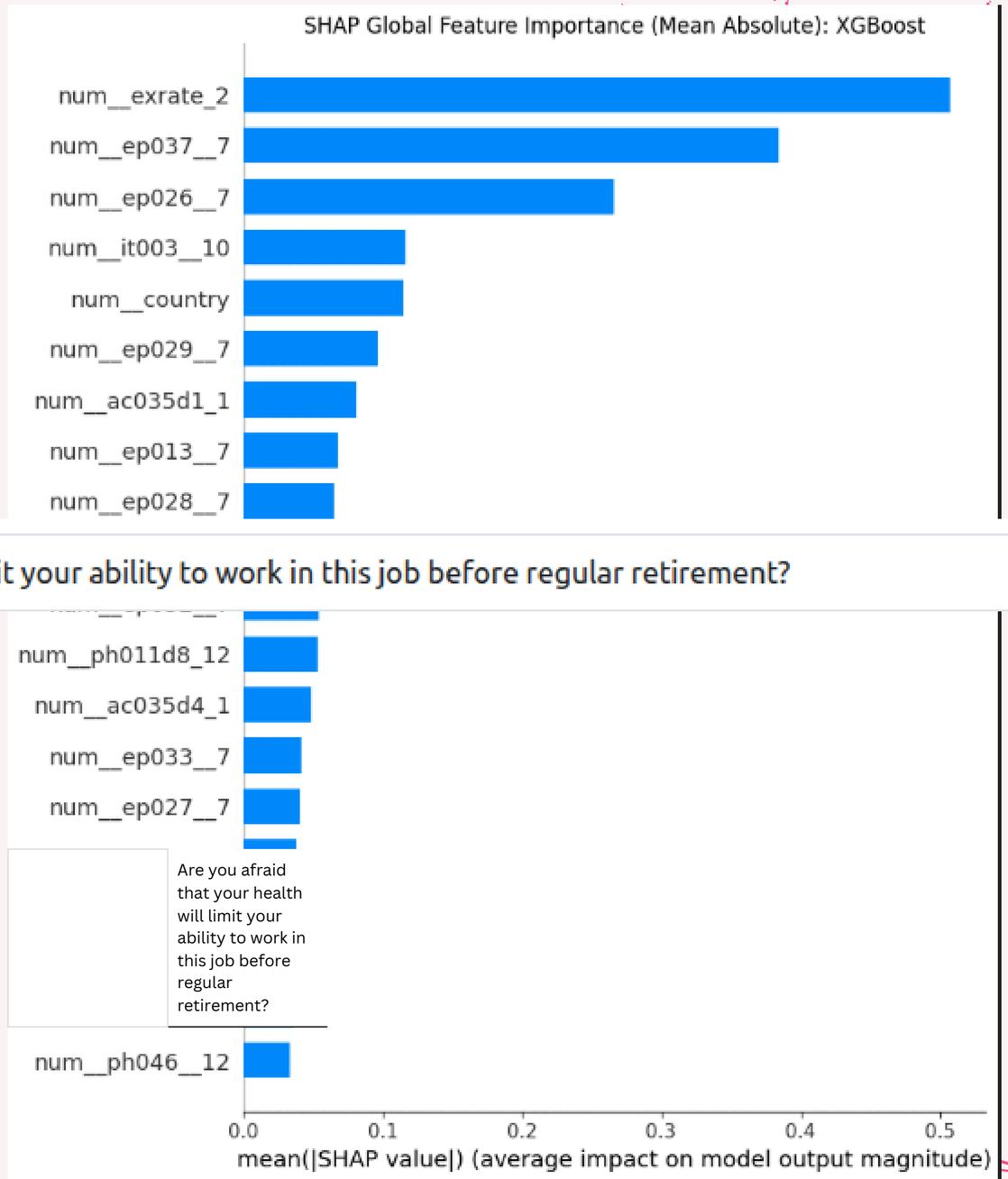

```
Finished LightGBM:
  Best CV ROC AUC: 0.7532
  Best Parameters: {'classifier__colsample_bytree': 0.8,

--- Test Set Evaluation for LightGBM (Best Estimator) --
  Test ROC AUC: 0.7410
  Test PR AUC: 0.7725

Classification Report (Test Set):
              precision    recall  f1-score   support

  Not Retire       0.67      0.67      0.67       135
      Retire       0.68      0.68      0.68       142

    accuracy                           0.68       277
   macro avg       0.67      0.67      0.67       277
weighted avg       0.68      0.68      0.68       277
```

RESULTS

	LightGBM	CatBoost	Support Vector Machine	MLP	XGBoost	Deep Neural Network
Precision-Recall	0.7725	0.7634	0.6724	0.6708	0.7607	0.9949
ROC AUC	0.7410	0.7338	0.6612	0.6927	0.7423	0.9954
Epochs/ Iterations	100/300	100/300	100/300	2000	100/300	30

CONCLUSION





EP037_AfraidHRet — Are you afraid that your health will limit your ability to work in this job before regular retirement?

THANK YOU

HTTPS://GITHUB.COM/METAPHYSICIST1/PREDICTING-RETIREMENT-AGE

AC - Accommodation
AS - Assets
BR - Behavioral risk
BS - Blood spots
CF - Cognitive function
CH - Children
CO - Consumption
DN - Demographics and networks
EP - Employment and pensions
EX - Expectations
FT - Financial transfers
GS - Gripstrength
HC - Health care
HH - Household income
HO - Housing
IT - Computer Use
IV - Interviewer
LI - Linking
MH - Mental health
PF - Peak FLow
PH - Physical health
SN - Social Network
SP - Social support
XT - End of life interview