Introduction to AI

Victor Abia Alonso

January 14, 2024

Student Number: 230061642

GitHub Repository: Click Here NOT DONE

1 Using machine learning models to understand subjective wellbeing.

REPORT STRUCTURE. Happiness. blablabla Life satisfaction. Happiness is one of the most important topics for modern humans, and it is usually considered to be one of the goals of life. ((At its core its about... and is understood as life satisfaction. KEY INFO.)). As contemporary morals keep shift towards focusing more on the feelings of people [**?**], the available data about mental health topics and subjective well being increases [1]. Still, our ability to model wellbeing is limited. Thus, In this article, I will explore the predicting power of ML approaches on wellbeing, replicating and expanding on the findings Oparina et Al [2].

The dataset The dataset I'm using for this task is UK Longitudinal Household Survey (UKHLS). Specifically, I worked over the 10th Wave of surveys (the "jkl" wave) which englobe the years 2018, 2019 and the first half of 2020. The UKHLS, also known as "Understanding Society", is the most comprehensive household panel in the UK its used to inform public policy. It collects individual and household information including demographics health, education, income and social attitudes, which is key to enable academic research and to draw insights about the nature of UK Society. The focus of this analysis is to predict, and thus model, self-reported life satisfaction, which is one of the variables collected by the survey by asking participants what is their satisfaction with life overall on a scale from 1 to 7 (integers only).

The processing of the dataset resulted challenging to the large amount of initial features 1,853 -each of them representing some information about the respondent- which were encoded in code format like *jkl_sato* (life satisfaction) and explained along with the meaning of their possible values, in a 300 page dictionary. Additionally, the dataset didn't include any indicator of which variables were numeric (e.g. age, income, life satisfaction) and which ones where categorical (e.g. voting preferences, education, race), because all data was stored as integers which could have a specific meaning as a label specified in the dictionary. As such, finding the categorical values was done by reading through all the relevant features.

Feature Selection Despite the large number of features, there where a lot of missing values as many of the variables represented placeholder attributes like *jkl_respchild16* i.e. whether the 16th person the respondent is responsible of was mentioned in their answer. Still, there were different types of missing values specified by different negative integers which still held some significant information. The labels -9 (*Missing*) and -8 (*Not applicable* are significantly different to -2 (*Refusal*), as the formers refer to the way the survey was set up while the later conveys information about the respondent, which could have predictive power. Still, all missing values where handled in the same way, but the analysis could potentially be improved by utilizing a sophisticated method that distinguishes among the different missing values. As most of the columns contained missing values, I run a loop to see how many columns would remain if I dropped the ones containing with a high rate of missing values [**?**], which is problematic because each column should be informative. Most of the columns with high missing values have almost all values missing, meaning that the difference between 70% of missing values and 25%, is not that big. In the end, I chose to stay with columns with at most 25% of missing values, which drop my dataset from 1,853 to 431 features.

I also dropped 27 columns closely related to life satisfaction to prevent data leakage. (TO-DO: Data leakage is the inference form non-target columns... the idea is that they are also the kind of columns the analysis wants to model and predict, rather than use for prediction). In this case, these dropped columns where survey questions about self-reported satisfaction on a specific life aspect like health or job, questions from the General Health Questionnaire (GHQ) (like "capable of making decision", "playing a useful role" and "enjoy day-to-day activities") and mental health related questions like having depression or anxiety. Additionally, I dropped survey related questions like identification of the respondent, date of the interview, how answers where collected, or survey-check variables. This intensive work of filtering, survey-specific respondent-independent variables, is the main difference of my analysis and the Oparina et Al [2] one, and probably the main reason of the relative improvement of performance (check whether more messy features can actually be bad)

Threshold (%)	Columns Kept	% of Total Columns
99	1194	64.44
95	827	44.63
90	665	35.89
75	582	31.41
60	568	30.65
50	517	27.90
40	449	24.23
30	440	23.75
25	431	23.26

Table 1: Comparison of Column Retention at Different Missing Value Thresholds

Feature processing. In order to train a machine learning model, features are usually scaled because most models tend to be bias for higher-magnitude features, which is irrelevant as magnitude is a consequence of the specific units of the feature (16000cm is more salient than 0,16km). All numerical features were normalized to have mean of 0 and variance of 1 because Linear Regression (model 1) benefits from standardization rather than MiniMax scaling, and Random Forest (model 2) is indifferent; missing values were imputed the mean value. However, 11 features related to income where scaled after taking logarithms, to avoid their meaning being destroyed by the nature of the heavy tail distribution. Categorical features where found by looking, one-by-one, all the columns with more than 2 unique non-negative values, and 11 of these were found related to voting preferences, education and family relations; curiously, race and origin related variables had too many missing values. Categorical variables where transformed into dummy variables (i.e. one-hot vectors) preserving the meaning of each label, which would be perturbed by the cardinality of numerical variables; missing values where given their own class.

The Machine Learning problem. The analysis consists on designing ML models that are able to predict the self-reported life satisfaction of individuals, which is a metric of subjective wellbeing and very close to what is usually considered happiness. The idea is to use the the information of a person's life encapsulated in the 300 used features to predict the life satisfaction level on a scale from 1 to 7. The analysis is thought as a regression problem due to the ordinal nature of the target variable, which despite being conceived as discrete levels of life satisfaction for data collection purposes, it represent an underlying continuous sentiment of oneself. (this doesnt mean it reflects a linear reality of sentiment, but rather ordinality). Posing it as a regression problem, allows the use of a variety of evaluation functions like MSE and to interpret the output of the model as a one dimensional variable, instead of as 7 different classes (one for each level of satisfaction) if it was posed as a classification problem. While a classification-problem apporach would struggle in this case due to the class imbalance of the dataset -most respondants answer 6 out of 7-, a regression approach doesn't struggle with that as treats the variable.... and loss functions based on distance like MSE or RMSE can handle this. Additionally, I dropped all the rows that had a missing value on the life satisfaction column. The target feature was also scaled to be between 0 and 1, but I will refer to it as its original labels when mentioning them.

PLOT GRAPH OR TABLE WITH THE CLASS IMBALANCE
$\{7.0 : 11.68, 6.0 : 43.45, 5.0 : 20.41, 4.0 : 11.0, 3.0 : 7.73, 2.0 : 3.96, 1.0 : 1.77\}$

[It will involve some reading and that's OK :), but please do it BBY Also the] The loss function utilized to train each model was the Mean Squared Error (MSE), which averages the square differences between the predicted and actual values for every training example as follows:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

where y_i is the actual value, $\hat{y}_i$ is the predicted value, and n is the number of observations. MSE is particularly suitable for this problem as it penalizes larger errors more severely by squaring them, which helps prevent the model from overlooking extreme low values such as 1 or 2. However, MSE primarily reflects prediction accuracy and does not directly indicate how well the model explains the

variance in the target variable. This aspect is crucially captured by the coefficient of determination, or R^2, which quantifies the proportion of the variance in the dependent variable that is predictable from the independent variables. As such $R^2 \in [0, 1]$, and it is calculated as:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \overline{y})^2}$$

where $\overline{y}$ is the mean of the actual values. A model that seeks to explain life satisfaction requires not just a low loss but also a high R^2, to ensure that the model explains the variation in the life satisfaction values. The reason R^2 is not used as a loss function for training is because it focuses on the overall distribution of data rather than specific data points, making it less intuitive as a loss function. Additionally, R^2 is not differentiable, which poses a challenge for gradient-based optimization methods commonly employed in many machine learning models used in this analysis. Also, instead of MSE, Root Mean Square Error (RMSE), which is simply the square root of MSE, will sometimes be used in this analysis to compare loss values for presentation purposes.

To illustrate these metrics, let's consider as a baseline example a model that always predicts the average of the data, in this case, always predict 5.67 (WRITE THE REAL NUMBER) regardless of the input variables. The loss of this classifier will be low (WRITE THE REAL NUMBER), as most of the points are pretty close to the mean. However, looking at the formula of R^2, the numerator -also called Sum of Squared Residuals (SSR)- and the denominator -also called Total Sum of Squares (TSS)- are equal. Thus $R^2 = 0$, which means that it is not capturing any of the variance of the life satisfaction. Curiously, R^2 can be negative if $SSR > SST$, which means that the model's error are larger than the intrinsic variance of the model, which means that it's a poor, perhaps extremely overfitted model.

In this analysis, I will mainly consider the linear regression model as a baseline and two tree-ensemble models: Random Forest and XGBoost. Additionally, I will touch on other ML models and consider their suitability for this problem. The dataset is split between training dataset -composed by 25,000 examples (i.e. 80% of all respondents)-, which is used to update the parameters of the model, and the testing dataset -composed by 8,000 examples (i.e. 20% of all respondents)-, which is used to calculate the MSE and R^2 metrics; these sets are chosen randomly.

Linear Regression.
Linear regression is a fundamental statistical technique that models the dependent (target) variable as a linear combination of the independent variables (predictors). The relationship is expressed by the formula:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n + \epsilon$$

where y represents the predicted target, $x_1, x_2, \ldots, x_n$ are the independent variables, $\beta_1, \beta_2, \ldots, \beta_n$ are their coefficients, β_0 is the intercept, and ϵ is the error term. This model is particularly useful for integrating multiple potential predictors in a simple and interpretable manner. The magnitude of the coefficient for each predictor variable represents the strength of its association with the target, and the intercept is the baseline predictor if all the variables were set to zero.

Calculating the parameters of a model to optimally fit the data is a key question in machine learning. In linear regression, this involves finding the coefficients $\beta_0, \beta_1, \ldots$ such that the loss (MSE in this analysis) between the predicted and target variable is minimized. Analytically, this can be achieved using the matrix normal method, which involves setting the derivative of the loss function to zero and solving for the coefficients. This method is expressed as:

$$\hat{\beta} = (X^T X)^{-1} X^T y$$

where X is the matrix of input features, y is the vector of target values, and $\hat{\beta}$ is the vector of the optimal coefficients. However, this analytical approach is not common for most machine learning models, which typically require iterative methods like gradient descent to approximate the optimal parameters and are subject to the risk of getting stuck in local minima. However, the analytical method can be computationally intensive for large numbers of predictors, as the resources needed to compute the inverse matrix grow cubically with the size of the input.

The most common numerical method to fit a multivariate linear regression model is gradient descent. This method updates each coefficient β_j in the direction that minimizes the loss function, by the update rule:

$$\beta_j := \beta_j - \alpha \frac{\partial}{\partial \beta_j} \text{MSE}(\beta)$$

where α is the learning rate, and the derivative $\frac{\partial}{\partial \beta_j}\text{MSE}(\beta)$ represents the partial derivative of the MSE with respect to the coefficient β_j. For linear regression, gradient descent methods also result in finding the global minimum fora sufficiently small α because the loss function is quadratic and thus convex.

The results using both methods are basically identical, with a final loss of NUMBER and coefficient of R^2 =?? REAL VALUE (0.4). This means that the linear model is able to explain 40% of the variation of life satisfaction among individuals by weighting the predictive variables. This may seem small, but considering the subjective nature of the traget variable, opposed to the objective facts used to predict it, it's significant. Additionally is one of the best performing methods we'll see, and is the standard statistical analysis for subjective wellbeing. However, we'll use ML techniques which could be more sophisticated than a purely linear regression and could potentially tackle more sophisticated patterns.

Yet, by looking at the magnitude of the coefficient, we can see what factors are, according to the linear model, more significant for predicting life satisfaction among participants. This suggests that the NAME THEM, PERHAPHS BRIEF EXPLANATION.

Random Forest.
The other principal method utilized in this analysis is Random Forest, an ensemble machine learning technique. This approach involves training multiple decision trees -over new datasets sampled with replacement (bootstrapped) from the original dataset- and then aggregating their predictions, typically by averaging. Ensemble strategies, sometimes likened to the 'wisdom of the crowds', capitalizes on the premise that while each individual model captures a unique and insightful aspect of the data, it also inherently carries its own noise and biases. As we integrate the predictions from multiple models, the idiosyncratic noise and biases of each tend to counterbalance and mitigate one another, allowing the consistent, true patterns within the data to emerge more prominently.

Decision trees are hierarchical, tree-like models that employ conditional statements at each node to reach a final leaf or decision. When used for regression, they group training examples by the similarity in variance of the output, using specific features as criteria. At each node, the model evaluates all possible divisions of the dataset by considering every potential threshold for each feature. The optimal threshold is selected based on its ability to minimize the variance resulting from the split. Through successive splits, training examples are eventually grouped into leaves, and the average value of the examples in each leaf then represents the tree's prediction for any new example that arrives at that leaf. (Decision trees always achieve 0 loss and R2 of 1 in the training set)

In a Random Forest classifier, while all trees may share certain characteristics such as maximum depth or nodes, they differ in the specific features they consider at each split. The number of features examined at each node is only a small subset of all available ones, so by randomly selecting which features are considered at each node, each decision tree becomes significantly distinct from the others. This is because the trees might not always choose the most variance-reducing feature as it may not be available from the subset of options. This method introduces sufficient variability among the trees, so that when predictions are averaged, the random forest model is robust and accurate, capturing a wider array of patterns and relationships in the data.

The results were close to the linear classifier, with a final loss of NUMBER and coefficient of R^2 =?? REAL VALUE (0.4). It is a model specifically designed to reduced the variance of the target variable, and it got a significant improvement over the other methods which at most were able to explain 37% of the variance ($R^2 = 0.37$). It also show an improvement over the performance of any decision tree used for regression, even after finetuning to find the optimal hyperparameters, which highlights the benefits of ensemble methods. However, it doesn't surpass the baseline performance of linear regression, despite the added complexity of combining non-linear decision trees. Now, in order to look at the most

significant factors predicting life satisfaction by the Random Forest model, we can look at the nodes of the trees that reduced the variance more. This apprach suggest that... NAME THEM, PERHAPHS BRIEF EXPLANATION (idk if this is the right method).

Other models.
Given the similar, and somewhat limited ($R^2 \approx 0.4$) results of the previous two models, I decided to explore the performance of other common machine learning model including neural networks, support vector machines and K-Nearest Neighbour Regressor and decision trees. For each model, the optimal hyperparameters where found via finetuning, in order to compare the best performing models of each type.

Indivividual decission trees were trained for this regression task. The baseline decision tree model, without any constrain on depth or the minimum samples per leaf performed worse that the always-precting-mean model. This is because an unconstrained decission tree will always overfit the training data, as every new split will reduce the relative variance on each child leaf, and the model will end with the minimum number of samples per leaf. As such, it optains optimal performance on the training data with $MSE = 0$ and $R^2 = 1$, but very poor on the testing data, with $R^2 \approx 0$. By limiting the model, by max-depth=5, mi-samples-leaf=10, the performance improve drastically obtaining $R^2 = 0.359$. After finetuning, some facts were clear about the parameters. It was clear that limiting the number of features at each split for just one tree is detrimental always for its individual performance as the model cannot choose the best variance-minimizing feature. However Random Forest can effectively capitalize on this considering several trees and avereaging them. Also, regressor trees benefitted of choosing the minimum number of samples required to be a leaf node to its maximum value, suggesting its regularization effect to avoid overfitting and it can get until 200 minimum samples per leaf node without decreasing R2 performance. Additionally, optimal maximum depth is 10 which was an intermidiate value of the ones tried. TO CHAT GPT

Individual decision trees were utilized for this regression task. The baseline model, an unrestricted decision tree, underperformed compared to a simplistic model that always predicts the mean due to the inherent tendency of unconstrained decision trees to overfit training data. In an unconstrained scenario, each new split reduces the relative variance in each child node, leading the model to eventually reach the smallest permissible number of samples per leaf. This makes the model achieve perfect performance on the training data, with $MSE = 0$ and $R^2 = 1$, but exhibit poor generalization to the testing data, yielding $R^2 \approx 0$. However, by imposing constraints on the model, specifically setting a maximum depth of 5 and a minimum of 10 samples per leaf, the performance significantly improved, achieving an $R^2 = 0.359$.

Fine-tuning revealed several insights about the model's parameters. It became evident that limiting the number of features at each split adversely affects the performance of an individual tree, as it restricts the model's ability to select the most effective feature for minimizing variance. In contrast, a Random Forest, which combines multiple trees, can exploit this limitation by averaging their predictions. Furthermore, setting a high threshold for the minimum number of samples required at a leaf node proved beneficial. This regularization technique effectively prevents overfitting, maintaining robust R^2 performance even with up to 200 minimum samples per leaf node. Additionally, the optimal maximum depth was found to be 10, an intermediate value among those tested, balancing model complexity and overfitting.

TABLE

Support Vector Machines (SVMs) are a set of supervised learning methods used for classification, regression, and outliers detection. The basic SVM is a linear classifier that finds a hyperplane in the feature space that best separates two classes by maximizing the margin, which is the distance between the hyperplane and the nearest data points from both classes, known as the support vectors. For regression tasks, the Support Vector Regression (SVR) variant of SVM is used, which tries to fit the error within a certain threshold (known as the ϵ-tube) and minimizes this error. The complexity of solving an SVM problem typically involves quadratic programming (QP) problems, with a computational complexity between $O(n^2)$ and $O(n^3)$, where n is the number of samples. In this case, for a

training dataset with 24,735 samples, the SVM algorithm's runtime exceeded two days before I decided to stopp it, underscoring the complexity and computational demands of SVMs on large datasets.

However, the computational burden of training an SVM can be mitigated by reducing the number of features in the dataset. Reducing the feature set through Principal Component Analysis (PCA), shortens the quadratic programming equations and the necessary training time. PCA involves calculating the singular value decomposition of the training data matrix, which decomposes the original feature vector into a linear combination of orthogonal singular vectors, also known as principal components. These principal components are ordered based on the variance they capture from the dataset, with the first component accounting for the most variance and each subsequent component accounting for progressively less. By selecting a fixed number of principal components, one can effectively reduce the dimensionality of the dataset with the principal components serving as the new, reduced features, each representing a training example. To investigate this approach, the SVR was trained on various reduced datasets with associated to increasing number of principal components, enabling an exploration of performance changes relative to the reduced feature set. Results are shown in the table along with the cumulative varaiance of the features that the principal components where able to capture.

PCs	**Cum. Var. (%)**	**R2 (SVR)**	**RMSE (SVR)**	**R2/Cum. Var.**
2	10.4	0.034	0.997	0.33
3	13.7	0.197	0.909	1.44
4	15.7	0.246	0.881	1.57
5	17.4	0.264	0.871	1.52
10	22.6	0.280	0.861	1.24
30	35.0	0.348	0.820	0.99
50	43.9	0.345	0.821	0.79
100	59.9	0.354	0.815	0.59

Table 2: Performance Metrics at Varying Principal Components

It is an intriguing observation that the R^2 value, representing the explained variance of the output data by the SVR model, can sometimes exceed the cumulative variance of the input data captured by the principal components. This phenomenon can occur due to the distinct nature of these two metrics. While PCA, a linear technique, captures the variance in the input features, it might not effectively represent non-linear relationships and interaction effects between features. On the contrary, SVR, particularly with non-linear kernels, can capture these complex relationships, leading to a higher R^2 value. Additionally, the principal components selected, though representing a smaller fraction of total variance, may contain critical information that is highly predictive of the output while the remaining variance may be noise, thereby enhancing the model's performance despite the reduced dimensionality.

The K-Nearest Neighbors (KNN) Regressor is a non-parametric, instance-based learning algorithm used in regression tasks which operates by finding the 'k' closest training examples in the feature space, and makes predictions based on the average output of these neighbors. The simplicity of the KNN algorithm lies in its use of a single key hyperparameter, the number of neighbors 'k' which dictates the number of nearest neighbors to consider when making predictions, and its optimal value is crucial for the model's performance. In our case, the KNN Regressor was fine-tuned showing that the higher it was, the better, without showcasing overfitting as its magnitude increased ($k = 50$ was similar to $k = 100$). It reached a maximum performance with an R^2 score of 0.179.

Neural networks, brief explanation, capitalise on provideing higher level representation on each layer. They are subsimbolic difficult to interpret systems that show amzing efficiency on computer vision and sequencial modelling task and have inspired a recolution of AI via deep learning. They consist on a sequence of layers with some neurons that are fully conected to each other in a linear way (described by a matrix of weights that is multiplyied to the output vector of the previous layer) and then an activation function which adds complexity to the model by enabling non-linear representations. Regularizations parameters can be added to avoid overfitting, in thsi case we use dropout with a rate of 0.3 or 0.5. The imput layer of the network is 447, the number of features in the dataset, and the output layer has just one neuron, representing the final prediction used for regression. The hidden

layers depend on the architecture, which was on of the hyperparameters tried, along with the learning rate and the dropout rate. The optimal set of hyperparameters consited on and reach a performance of Architecture (256, 128, 64), Dropout: 0.5, Learning Rate: 0.0005 MSE: 0.6583127332455632, RMSE: 0.8113647350270796, R2: 0.36091020674922936.

Neural networks, at their core, excel in providing higher-level representations through each successive layer. Characterized as sub-symbolic and often challenging to interpret, these systems have demonstrated remarkable efficiency in tasks such as computer vision and sequential modeling, fueling a revolution in artificial intelligence via deep learning. A neural network is composed of a series of layers, each containing neurons that are fully connected through a linear transformation which is represented by a weight matrix applied to the output vector of the preceding layer. The incorporation of an activation function at each layer adds complexity, enabling the network to capture non-linear relationships in the data. To mitigate the risk of overfitting, regularization techniques such as dropout are employed in this analysis, with dropout rates of 0.3 and 0.5. The input layer of our neural network corresponds to the number of features in the dataset, totaling 447, while the output layer consists of a single neuron, which generates the regression prediction. The architecture of the hidden layers, a critical hyperparameter, was varied in our experiments, alongside the learning rate and dropout rate. The optimal set of hyperparameters was determined to be an architecture with layers of 256, 128, and 64 neurons, a dropout rate of 0.5, and a learning rate of 0.0005. This configuration achieved a Mean Squared Error (MSE) of 0.658, a Root Mean Squared Error (RMSE) of 0.811, and an R^2 score of 0.361, highlighting the effectiveness of this particular neural network setup in our regression task.

Model	**R2 (SVR)**	**RMSE (SVR)**		
2	10.4	0.034	0.997	0.33
3	13.7	0.197	0.909	1.44
4	15.7	0.246	0.881	1.57
5	17.4	0.264	0.871	1.52
10	22.6	0.280	0.861	1.24
30	35.0	0.348	0.820	0.99
50	43.9	0.345	0.821	0.79
100	59.9	0.354	0.815	0.59

Table 3: Final results table

Explaination of the models (just to show off, optional) and the results, linking it to the models themselves.

Fine tune the hyperparameters (just for random forest because linear regression...)

Conclusion

Possible ADD: train XGBoost algorithm... Possible ADD: train random forest with extra bootstrapping from 1, 2 and...

Limitation from simplicity.

Explore other models (not Random Forest, or GBostX) and introduce them showing off ;). NN, SVM, DecisionTree, KNN Regressor, any other?. Expalin results and why they dont work, perhaps. Think about whether finetuning makes sense just to make a point, mention in which that makes sense (not LinearRegression).

Potential improvement: train model challenging class imbalance.

STILL TO DO (code) Find the most significant features for Linear (graphs? covariance matrix?) Train neural network (keras, tf), and do short finetuning perhaps. Any other model?? Hyperparameters? Do dimensionality reduction? on SVM Train the random forest. Hyperparameters. What features are important? (graphs? covariance matrix?) Improvement: train each tree with the same number of 1s, 2s, 3s, .., 7s

References

[1] Ed Diener, Shigehiro Oishi, and Louis Tay. Advances in subjective well-being research. *Nature Human Behaviour*, 2(4):253–260, 2018.

[2] Ekaterina Oparina, Caspar Kaiser, Niccolò Gentile, Alexandre Tkatchenko, Andrew E. Clark, Jan-Emmanuel De Neve, and Conchita D'Ambrosio. Human wellbeing and machine learning, 2022.