**Daniel Tomaro**

**Task 1. Load the dataset, do basic data pre-processing and split the dataset.**

The Pandas library was loaded to inspect [**Life Expectancy.csv**](https://lms.latrobe.edu.au/pluginfile.php/10728659/mod_page/content/14/Life%20Expectancy.numbers) which was collected and provided by the World Health Organisation (WHO). Using **pandas** and **pd.read\_csv().** **Len(dataset)** was used to understand the number of records present which was 2928 with 20 variables including life expectancy. **Dataset.head()** provides a preliminary look at the data's structure and content with only one variable being nominal.

Using **dataset.isna().sum()** to check for missing values (19 in three separate columns), they were removed for simplicity by using **dataset.dropna(). Dataset.duplicated().any()**, ensures data integrity by checking for duplication. The categorical **Status** column was converted to numerical values by replacing **developed with 1** and **developing with 0**. **dataset.describe()** and **dataset.shape** help understand the final dimensions and structure being 2909 columns and 20 rows after pre-processing.

A **NumPy** array is formed, with **X** representing the features and **Y** the target variable. To partition the data into training and testing sets, we employ the **train\_test\_split** function from **sklearn.model\_selection**, allocating 10% of the data for testing and 90% for training.

Normalization was applied using **MinMaxScaler** from **sklearn.preprocessing**. The scaler is first fitted to the training data with **norm = MinMaxScaler().fit(X\_train)**, establishing the scaling parameters based on the training set. We then transform both the training and testing datasets as this normalization process rescales feature values to a range between 0 and 1, ensuring fairness and enhancing the model.

**Task 2. Train and evaluate the three regression models on the training set with a cross-validation method, optimise the models and evaluate models on the test set.**

Abdullah (2021) explains SVM as a powerful classification model that finds the optimal hyperplane to separate data points into different classes by maximizing the margin between them. It can handle both linear and non-linear relationships using kernel functions, which transform the input data into higher dimensions where a linear separation is possible. This is primarily used for classification.

Maulaud (2020) states Linear Regression is a statistical method used to model the relationship between a dependent variable and one or more independent variables by fitting a line or hyperplane that minimizes the sum of squared errors between the predicted and actual values. It assumes a linear relationship, independence of observations, homoscedasticity (constant variance of errors), and normally distributed residuals. This is primarily used for prediction and estimation.

Zhang (2016) in his research found KNN is a simple, non-parametric algorithm that makes predictions by identifying the 'k' closest data points in the training set and using their values (or majority class in classification tasks) to determine the outcome for a new data point. It heavily relies on distance metrics like Euclidean distance and is sensitive to the scale of features, making normalization important. KNN can be used for both prediction and estimation.

The **LinearRegression** class was imported from the **sklearn.linear\_model** module. The obtained R² score for the linear regression model indicated that approximately 83.24% of the variance in the dependent variable can be explained by the independent variables included in the model. This score suggests a reasonably strong fit of the linear regression model to the data.

A Support Vector Regressor model from **scikit-learn's svm** module was then employed. The R² score obtained for the Support Vector Regressor model was 86.74%. The higher R² score compared to linear regression (0.8324) suggests that the SVM model might better capture the underlying patterns in the data.

In the 5-Fold Cross-Validation the dataset is randomly divided into five equally sized subsets or "folds." The model is trained on four of these folds and tested on the remaining fold. This process is repeated five times, with each fold being used as the test set exactly once. The results from each of the five tests are then averaged to provide a more robust estimate of the model's performance. This technique helps in minimizing the bias that can occur from relying on a single train-test split and provides a better indication of how the model will generalize to unseen data. The dataset was shuffled prior to splitting into folds to ensure randomness, using a random seed of **123** for reproducibility. The **KFold** class from **sklearn.model\_selection** was utilized and a default Linear Regression model **lr = LinearRegression()** was instantiated, and the **cross\_val\_score** function was employed to perform cross-validation.

The average R² score for the Linear Regression model was approximately 83.8%, indicating a consistent level of predictive performance across the different folds. To optimize the performance of the Linear Regression model, we use **GridSearchCV** from **sklearn.model\_selection** to perform an exhaustive search over specified hyperparameters. In this case, we are tuning two hyperparameters: **fit\_intercept** and **positive**. The **fit\_intercept** parameter determines whether to calculate the intercept for the model or not. The **positive** parameter, when set to **True**, constrains the model coefficients to be non-negative. By setting up a grid of possible values for these parameters, **GridSearchCV** evaluates all combinations using cross-validation with the previously defined **kfold** strategy. The result of 83.76% reflects the highest mean cross-validated score achieved during the grid search.

The same was done for the Support Vector Machine (SVM) model to explore a range of hyperparameters, including the kernel types, regularization parameter (with values 1 and 5), and the kernel coefficient (**Gamma).** This method randomly samples from the hyperparameter space for a specified number of iterations, using cross-validation to evaluate each combination. The best performing set of hyperparameters produce a value of89.69%**.** Subsequently, **GridSearchCV** is applied to refine the search by exhaustively testing all specified hyperparameter combinations over the same grid of parameters. This thorough search process, performed with **n\_jobs=-1** to utilize all available processors.

Evaluating the optimized Linear Regression model on the testing resulted in an R² indicating that the model explains approximately 83.2% of the variance in the test data. Additionally, the best hyperparameters identified for the Linear Regression model are **fit\_intercept=True** and **positive=False**. Whereas evaluating the optimized Support Vector Machine (SVM) model on the testing set the R² score was 0.912, which is a significant improvement over the initial R² score of 0.867. This enhancement reflects the effectiveness of the hyperparameter tuning process. The best parameters for the SVM model are found to be **C=5`, gamma=scale**, and **kernel=rbf**, suggesting a more flexible and complex model configuration that improves performance.

**References**

* Abdullah, D.M. and Abdulazeez, A.M., 2021. Machine learning applications based on SVM classification a review. *Qubahan Academic Journal*, *1*(2), pp.81-90.
* Maulud, D. and Abdulazeez, A.M., 2020. A review on linear regression comprehensive in machine learning. *Journal of Applied Science and Technology Trends*, *1*(2), pp.140-147.
* Zhang, Z., 2016. Introduction to machine learning: k-nearest neighbors. *Annals of translational medicine*, *4*(11).