Income Prediction using Machine Learning

Data Overview and Preprocessing

The income dataset provided has a total of 26215 rows and 10 columns, with the data variable types being objects or integers. Therefore, there was no need to change the integer type for the variables besides the categorical data. The first step taken was to identify whether there were any missing values by finding the sum of missing data within each column. From this, we can see that there are 1396 missing values under the column 'workclass' and 1401 under the column 'occupation'. I chose to delete these missing values after doing a quick analysis, as there was a strong correlation between the different occupations and workclass, on their respective income. Hence, even though removing 1401 missing values are significant for the dataset, simply using the most occurring (mode) workclass and occupation to handle the missing values will greatly skew the data. By removing these missing values, 1401 rows were deleted and we are left with 24814 rows. The next step was to identify whether there were any duplicated inputs of data, and remove them accordingly. It is vital to remove duplicates as it negatively affects the performance and accuracy of the model. Using a similar method to dropping missing values, the dataset is now left with 21537 rows.

After this, columns with categorical variables were handled as they were non numeric. Categories are not compatible with logistic regression and k-mean clustering algorithms as it requires numeric inputs. The column education values were able to be replaced effectively with meaningful numeric values. The column sex was also converted into numbers and dummy coding was used to deal with the columns workclass, marital-status, occupation, relationship and race. Following these transformations, the dataset now has 21537 rows and 41 columns.

The next step was to split the dataset into the target and predictors. The predictors, X, were set for the columns 2-41 and the target (income), y, was the first column. The dataset was then split with 10% used for testing and 90% used for training to measure the generalisation capacity of the model, since the testing dataset is used to evaluate the model's performance on unseen data. Following this, normalisation was applied to ensure that training is more stable and improve convergence and performance of the model, as machine learning algorithms are sensitive to the scale of the input features. This is achieved by mapping the minimum and maximum values of each predictor, scaling the predictors to a certain range using the MinMaxScaler accordingly.

Supervised Learning - Logistic Regression and Support Vector Machine (SVM)

Firstly, we will be using Logistic Regression (LogR) and Support Vector Machines (SVM), both being supervised machine learning algorithms that are used for classification to evaluate the features. LogR is used in binary classification to determine whether the output variable, being income, is 0 (less than or equal to 50k) or 1 (more than 50k). This model would thus learn the association between the input features (education, marital-status, occupation, etc) and the output (income). It would therefore predict the probability for a given feature and its subsequent classification into one of the two output classes.

On the other hand, SVM is used to find the decision boundary (known as the hyperplane) and maximise the margin which separates the data into the two different classes of income. The margin is the distance between the data and the hyperplane and the best result is known as the maximum marginal Hyperplane (MMH). Getting the MMH is the ultimate goal for SVM classification as it provides better separation of class, improves generalisation and makes it more robust to noise.

Please refer below to the table for the results after defining the two models and using 10-fold cross validation on the defaults settings.

Model	Average Accuracy Score: Training Dataset	Testing Accuracy
Logistic Regression	0.8051391485722164	0.8115134633240483
Support Vector Machine	0.7971934508175302	0.7961931290622098

10-fold cross-validation was used to train the data which increases the statistical significance and generalisation of the data as each fold consists of different parts of the data. Average accuracy scores in training and from testing were very similar for each model. Without any finetuning, the logistic regression model is slightly more accurate in both datasets for training (0.8%) and testing (1.5%) over SVM.

Fine Tuning - Logistic Regression

To fine tune the logistic regression model, I created 3 grid search sets of parameters to best use the different solvers applicable in logistic regression. This worked to account for the different penalty values of each hyperparameter and allow easy documentation of the varying results. The results for the fine tuning of the logistic regression model are shown in the table below.

Model - Log Regression	Best Score (Training)	Accuracy in Testing	Parameters
Parameters 1	0.8053971731196754	0.8105849582172702	{'C': 1, 'penalty': 'l2', 'solver': 'liblinear'}
Parameters 2	0.805242347746623	0.8115134633240483	{'C': 1, 'penalty': 'l2', 'solver': 'sag'}
Parameters 3	0.8051907481594196	0.8115134633240483	{'C': 1, 'l1_ratio': 0, 'penalty': 'elasticnet', 'solver': 'saga'}

Judging by the testing accuracy of the different parameters, the different solvers did not have an effect on the performance of the model. In fact, using the liblinear solver proved to be less effective, as the model may have been mildly overfitting on the training data and thus slightly decreasing testing accuracy. The testing accuracy of the fine-tuned model with the solver sag and saga was identical to the results obtained pre fine-tuning, however, there was a slight increase of 0.01% and 0.005% of the training score respectively. Since testing accuracy is a more significant means of evaluating a model's performance and generalisability, the impact of the very slight change in training accuracy is up for debate on the effectiveness of fine tuning in this situation. I chose to continue the analysis with parameters 2, as it had the higher training score.

Fine Tuning - Support Vector Machine

Due to the expensiveness of the computational power, it was difficult to try and fit different kernels for training the SVM model in the same grid search. Therefore I ran multiple different iterations of the grid search by changing the kernel value to test different regularisation parameters (C), degree and gamma values. The results are shown in the table below.

Different Kernel Run	Best Score (Training)	Accuracy in Testing	Parameters
Linear	0.79920676611895	0.7999071494893222	{'C': 10, 'degree': 3, 'gamma': 'auto', 'kernel': 'linear', 'max_iter': 20000}
Poly	0.804003664927875	0.8040854224698236	{'C': 1, 'degree': 8, 'gamma': 'scale', 'kernel': 'poly', 'max_iter': 20000}
Rbf	0.806892709582408	0.8068709377901578	{'C': 10, 'degree': 3, 'gamma': 'scale', 'kernel': 'rbf', 'max_iter': 20000}
Sigmoid	0.7998258281081766	0.8012999071494893	{'C': 10, 'degree': 3, 'gamma': 'auto', 'kernel': 'sigmoid', 'max_iter': 20000}

In an attempt to further optimise the model, C was changed to a value of 100 for rbf, with the results shown below. Although the training score decreased by 0.3%, there was an increase in the accuracy in the testing dataset of approximately 0.2%. Therefore, I chose to continue with the C value set as 100 for the fine-tuned model as testing accuracy is a more significant measure of the model's effectiveness.

Kernel	Best Score (Training)	Accuracy in Testing	Parameters
Rbf	0.8036426274861075	0.8087279480037141	{'C': 100, 'degree': 3, 'gamma': 'scale', 'kernel': 'rbf', 'max_iter': 20000}

The model that performed the best was the kernel rbf with a regularisation parameter value of 100. The training accuracy of the fine-tuned improved by approximately 0.64% compared to the default model. On the other hand, the testing accuracy of the fine-tuned model improved by approximately 1.25%.

Therefore, from fine-tuning, the SVM model showed a more significant increase in its accuracy across both the training and test set of data compared to logistic regression, which only displayed a slight increase in training average accuracy score. Despite this, when comparing the best results from both cases, logistic regression has a slightly higher training score of 0.16% and a higher testing accuracy of 0.46%. Hence, the results support that logistic regression using the default settings or with fine-tuning is slightly more accurate than SVM.

Model	Type	Best Score (Training)	Accuracy in Testing
Logistic Regression	Default settings	0.8051391485722164	0.8115134633240483
	Fine-tuning (sag)	0.805242347746623	0.8115134633240483
Support Vector Machine	Default settings	0.7971934508175302	0.7961931290622098
	Fine-tuning	0.8036426274861075	0.8087279480037141

Unsupervised Learning - K Means Clustering

The final part of the report was to apply K Means clustering on the data. As opposed to the two machine learning algorithms above, K Means is an unsupervised clustering algorithm. K Means is used to find the centre of the cluster and assigns data points to the nearest cluster centre. This works to partition the data in groups of equal variance, assigning it to only one of the clusters based on similarities of the features. For this specific dataset, I chose to use two clusters as there are only two classes of income (0 or 1). From the K Means model 10123 data samples were assigned to the cluster class 0 and 9260 data samples were assigned to the cluster class 1. From that, we then extracted a prototype from each cluster to observe the differences in the features. The prototype for each cluster are shown below:

Feature	1st Cluster Index Sample: 20243	2nd Cluster Count Index Sample: 19208
income	0	0
age	42	19
education	9	9
sex	0	0
hours-per-week	50	15
workclass_Federal-gov	0	0
workclass_Local-gov	0	0
workclass_Private	1	1
workclass_Self-emp-inc	0	0
workclass_Self-emp-not-inc	0	0
workclass_State-gov	0	0
workclass_Without-pay	0	0
marital-status_Married	0	0
marital-status_NotMarried	0	1
marital-status_Separated	1	0
marital-status_Widowed	0	0
occupation_Adm-clerical	0	0
occupation_Armed-Forces	0	0
occupation_Craft-repair	0	0
occupation_Exec-managerial	0	0
occupation_Farming-fishing	0	0
occupation_Handlers-cleaners	0	0
occupation_Machine-op-inspct	0	0
occupation_Other-service	0	1
occupation_Priv-house-serv	0	0
occupation_Prof-specialty	0	0
occupation_Protective-serv	0	0
occupation_Sales	0	0
occupation_Tech-support	0	0
occupation_Transport-moving	1	0
relationship_Husband	0	0
relationship_Not-in-family	1	0
relationship_Other-relative	0	0
relationship_Own-child	0	1
relationship_Unmarried	0	0
relationship_Wife	0	0
race_Amer-Indian-Eskimo	0	0
race_Asian-Pac-Islander	0	0
race_Black	0	0
race_Other	0	0
race_White	1	1

The differences between the two datasets are highlighted in yellow. Both cluster prototypes belonged to income class 0, which suggests the grouping of similar features from K Means did not accurately differentiate between the two income classes. Education, sex, work class and race were the same in both clusters. The differences observed between the two prototypes were age, hours worked, marital status, occupation and relationship.

Model	**Best Score (Training)**	**Accuracy in Testing**
K Means	0.7193932827735645	0.30362116991643456
Logistic Regression {'C': 1, 'penalty': 'l2', 'solver': 'sag'}	0.805242347746623	0.8115134633240483
SVM {'C': 100, 'degree': 3, 'gamma': 'scale', 'kernel': 'rbf', 'max_iter': 20000}	0.8036426274861075	0.8087279480037141

The accuracy of the clustering method based on the testing dataset showed to have lowest results out of the 3 models. Additionally the results from K Means clustering method shows signs of overfitting with a high training score and a significantly lower testing accuracy. This suggests the model performs poorly on unseen data and would be difficult to generalise and thus may be beneficial to perform fine-tuning. In conclusion, from the results outlined in the above table, the best model in terms of accuracy for the testing set was the logistic regression model, even without the necessary requirement of fine tuning.