Income Classification Using Census Data
Final Report
IDS 575 – Machine Learning and Stats

By Group 2
Anoop Gopalam (UIN: 676766001)
Anvesh Nadipelli (UIN: 651408842)
Reshmika Reddy (UIN: 661035619)
Shivangi Patel (UIN: 665544840)

Problem Statement and Motivation:

Declining economic mobility, regional income segregation, and adverse effects on the political influence of lower-income groups are all caused by rising economic inequality.

To determine a causal relationship, economists include data sets in their theoretical models, such as the Census Income Dataset. The expense of using these databases, however, is high. Income inequality is commonly measured based on household income estimates from the Current Population Survey (CPS), a survey carried out by the U.S. Census Bureau and the Bureau of Labor Statistics.

A specific subset of the population's opportunities is measured using their median salaries. For instance, people with high average incomes are more likely to succeed than those from lower socioeconomic backgrounds. According to this interpretation, disparities in median income and opportunity can coexist. In other words, the greater the wealth disparity, the greater the presence of deterministic exogenous causes outside the control of any one person.

To put the strategy above into practice, one needs to explicitly describe what variables to include in a statistical model given a data collection, such as CSP.

To categorize the population into different categories, we must divide it into distinct segments.

For instance, you might divide the population into groups according to parental occupation, property ownership, and educational attainment levels. But was this decision random?

Do these criteria provide enough information to draw any relevant conclusions?

In contrast, if one includes too many factors, then much of the inequality associated with circumstances outside an individual's control will be due to sampling error. Including only a few elements would result in underestimations of inequality of opportunity.

An economic model's internal development of parameters is possible with machine learning techniques. Machine learning methods can consider all factors outside of an individual's control as inputs, which then apply algorithms to categorize the population into different groups. As a result, the segmentation will use pure data analysis rather than random decisions made by a researcher.

In this research, we used the Income Census Data to predict, given a set of characteristics, whether the respondents' income is higher or less than $50,000. This classification can be applied to business applications, targeting customers, marketing campaigns, and expansions. By implementing Support Vector Machine (SVM), Logistic Regression, and Linear models, we will test the precision of several machine learning models.

Model accuracy, precision, recall, and F1 scores were the foundation for our performance analysis. We considered individual traits like age, worker class, education, enrollment status in postsecondary institutions, income, and gender to build our various models. We may then forecast this person's economic potential by considering the characteristics of their median income.

Related Work:

The nation obtains information about its people and economy through the Census Bureau, which gives the country pertinent population data.

We could comprehend several aspects of the CPS dataset thanks to an article on Analytics Vidhya about examining the income level of US Census data.

Georgetown University has taken on our goal of forecasting an income over or below $50,000. Although developing precise forecasting models is difficult, using Census data makes this possible. They employ a dataset with 14 attributes and around 50,000 cases, breaking machine learning into various components. They had a few issues with the datasets they were working with and eventually achieved an F1 score of.77. This article enables us to optimize the predictive models in our project. It is a terrific resource we can utilize as the foundation for our work.

Models:
We decided to construct our models to compete with one another. Our goal of this project is to specifically understand how LR competes with SVM and kNN competes with LVQ. We shall explore the optimal models and their parameters in this study.

Naïve Bayes:
Our only truly generative model was our baseline model, and its formulation may be viewed as a machine learning application of Bayes Rule (see left). Using Naive Bayes, we estimate the likelihood that a given instance belongs to class Y. We calculate the conditional probability for each class given the features of a particular instance and select the class probability with the highest value as the prediction for the instance. The following formulation can be considered for Navie Bayes:

$$P(y|X) = \frac{P(X|y) * P(y)}{P(X)} \qquad y = argmax_y[P(y) * \prod_{i=1}^{n} P(x_i|y)]$$

Logistic Regression:
To comprehend Logistic Regression, we must first comprehend Linear Regression. As illustrated below, in the traditional linear model, a vector of coefficients (represented by Theta /θ) is multiplied by a feature vector designated by our X superscript values.

$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \ldots + \theta_j x_j \quad = \theta^T x \quad = \begin{bmatrix} \theta_0 & \theta_1 & \ldots & \theta_j \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \ldots \\ x_j \end{bmatrix}$$

$$Where\ x_0 = 1$$

This yields our hθ value, which would be our prediction in the traditional linear regression situation; but, in the logistic regression scenario, this is merely the first half of the formulation, as we are attempting to develop a binary classification system. The thought process is to map the prediction from linear regression to a likelihood or probability value between zero and one. This is accomplished by the sigmoid function, which performs the mapping and modifies our hypothesis to the output of the vector multiplied by our features plugged into the sigmoid function. With the addition of a regularizer, class weights, and a stochastic average gradient loss function, this fundamental formulation served as the foundation for the project. We used lasso or L1 regression to boost robustness and send non-essential coefficients as zero. This decision influenced our loss function, as we had to adopt a variant that accommodated the non-smooth penalty term.

Support Vector Machine:
The foundation equation for SVM is fairly like our linear regression equation. In addition to learning our coefficients or theta parameters, we also learn a distinct intercept term, b. In Linear Regression, the initial theta or θ zero represents the intercept. The aim of SVM is to map our features into a higher-dimensional space in which we can easily distinguish our classes with a hyperplane; separating the intercept term is partially what

enables us to achieve this and is hence not included in the formulation. Following is the model's soft margin equation:

$$\left[\frac{1}{n}\sum_{i=1}^{n}\max\left(0,1-y_i(w\cdot x_i-b)\right)\right]+\lambda\|w\|^2$$

The soft-margin model would behave more like a hard-margin model if lambda was set to a very tiny number in the above formulation. In our modeling, we chose for an RBF kernel with a C value of 1 and a gamma of 0.01. Following cross-validation on several values to determine the ideal set for these data, we arrived at these. Another key distinction between SVM and Logistic Regression is that SVM may utilize alternative kernels to successfully adjust the decision boundary, but logistic regression can only search for the optimal boundary for the given data.

K-Nearest Neighbors:

Our first similarity classifier was kNN, and we chose Euclidean distance as our primary measure of similarity for this research. We considered using Manhattan distance, but since our data was already normalized, we decided to stick with Euclidean. An argument for choosing Manhattan instead would be that we have many binary or one-hot-encoded variables, and because of this or rather because the dimensionality is so high, Manhattan may be the best option. The formula for Euclidean distance can be thought of as follows: where each instance's difference is calculated as described above, and the class predicted for the next instance is the majority class within the K-features distance.

$$\text{Euclidean:} \qquad \sqrt{\sum_{i=1}^{k}(x_i-y_i)^2}$$

Learning Vector Quantization:

At first glance, it may appear that LVQ and kNN are essentially the same algorithm, but the primary difference between the two makes LVQ a much ‘bolder’ choice: whereas kNN must memorize or store the entire dataset in order to make predictions on new data, LVQ attempts to synthesize the dataset into a number of 'sufficient' codebook vectors. These vectors are believed to capture all pertinent data information without the burden of storing the complete dataset in memory! The formula for LVQ can be formulated as follows: Given several subclasses or neurons, we assign each to a class or label and utilize them to make predictions when presented with a new instance. We calculate the Euclidean distance from each occurrence to our unique data point and assign a weight to each instance; these weights are updated with each epoch and effectively assign each of the codebook vectors a relevance value. Once weight improvement is no longer significant or has reached a point where weight change is meaningless, or when the period limit is achieved, learning ceases and the algorithm is prepared to provide predictions.

Methods:

After reviewing relevant/supplementary information, we've decided to build the Nave Bayes, Logistic Regression, and SVM models from the ones we've learnt so far in the course. It was a no-brainer for us to choose Nave Bayes as our baseline, given that our dataset is sizable, and we can therefore create a decent approximation of the distribution. In terms of performance, it will likely be one of our lower-scoring models, but it provides us with a benchmark for determining what constitutes "excellent" in this dataset. We were also interested in the distinctions between logistic regression and support vector machine (SVM). Since the two methods are similar in their fundamental design, we sought to determine whether their performance was also comparable. To begin, we are considering comparing the performance of basic SVM and logistic regression classifiers and, after analyzing the respective base forms, utilizing the unique characteristics of each model. L1 regularization or Lasso regression is intended to make our model more resilient and identify crucial factors by

reducing coefficients of less significant variables to zero. By evaluating several SVM kernels that may or may not be more applicable for this dataset, we seek to improve model performance while gaining a deeper understanding of the algorithm. We intend to deal with the fact that our dataset is relatively unique and contains some NA values by utilizing kNN to impute variables. We would examine the performance of data with and without kNN imputations to determine whether they eventually contribute more noise or help clarify data relationships. In addition to using kNN as a classifier, we will add it to our arsenal of models. A member of our group stumbled into the idea of "Learning Vector Quantization," which is comparable to kNN and can be considered a tiny neural network. The technique offers a novel approach to the problem, which we believe may give an interesting contrast that will be enjoyable to apply. Finally, once we have reviewed a few clustering methods, we anticipate utilizing one of them to search for relationships in the data, maybe identifying unique groups. However, as we have not yet delved deeply into this topic, we are delaying the creation of plans.

In conclusion, the following algorithms will be implemented:
Nave Bayes (NB)
Logistic Regression (LR)
Support Vector Machine (SVM)
k-Nearest Neighbor (kNN)
Learning Vector Quantization (LVQ)

Evaluation Metrics:
Accuracy, precision, and AUC are often employed measures for evaluating the performance of a classification model.

Accuracy
The accuracy statistic measures the proportion of true predictions made by a model. It is determined by dividing the number of accurate forecasts by the total number of predictions. For instance, if a model makes 100 predictions and 75 of them are accurate, the model's accuracy is 75%.

Accuracy is a valuable metric since it gives a straightforward and intuitive way to evaluate a model's overall performance. However, if the dataset is unbalanced, it can be misleading because it does not account for the relative importance of the different groups.

Precision
Precision is a metric that indicates the proportion of a model's correct positive predictions. It is derived by dividing the number of accurate positive forecasts by the total number of positive forecasts. For instance, if a model generates 10 positive predictions and eight of them are accurate, the model's precision is 80%.

Precision is an important metric since it allows for the evaluation of a model's performance on the positive class. It is particularly beneficial in cases where minimizing the frequency of false positive predictions is crucial, such as in medical diagnosis and fraud detection.

Area Under the Curve
AUC (area under the curve) is a metric that gauges a model's ability to accurately differentiate between positive and negative classifications. It is the area under the receiver operating characteristic (ROC) curve, which is a plot of the true positive rate (TPR) vs the false positive rate (FPR).

AUC runs from 0 to 1, with 0.5 representing the expected performance of a random model and 1 representing the best performance achievable. A model with a high AUC value accurately distinguishes between positive and

negative classes and makes accurate predictions. A model with a low AUC value, on the other hand, is unable to effectively distinguish between positive and negative classes and does not make accurate predictions.

In conclusion, accuracy, precision, and AUC are regularly employed measures for evaluating the performance of a classification model. Accuracy measures the proportion of correct predictions made by a model, precision counts the proportion of correct positive predictions made by a model, and AUC indicates a model's ability to correctly distinguish between positive and negative classifications. These measurements provide

Fine Tuning of Logistic and Support Vector Machines – Grid Search:
Grid search is a method for hyperparameter optimization, which involves determining the optimal set of hyperparameters for a given machine learning model. Different sets of hyperparameters can result in drastically different model performance.

In the case of logistic regression and support vector machines (SVMs), grid search can be used to determine the optimal regularization strength and kernel type parameters.

To employ grid search for logistic regression or SVM, you would specify a range of searchable values for each hyperparameter. You may want to experiment with other regularization strength settings, such as 0.1, 1, and 10.

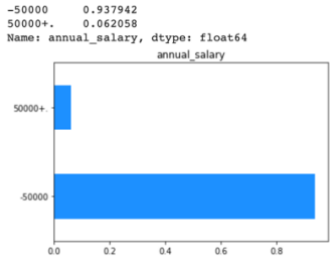
The grid search algorithm will then train a model with each hyperparameter combination and evaluate the performance of each model using a metric such as accuracy or F1 score. The set of hyperparameters that yields the highest performance will be chosen as the optimal set for the model.

Grid search is an effective method for determining the optimal hyperparameters for logistic regression and SVM models. It can help you improve the performance of your model and make predictions on new data more accurate.

Experimental Results:
After analyzing our data, we have prepared the following summary:

Dimensions of the whole Dataset (299,285 rows, 42 features)
Instructional Section
- 199523 Test Portion
- 99762 Training Portion
Types of Features:
- 29 "Object" and
- 13 "Numeric"

Initial Label Imbalance:



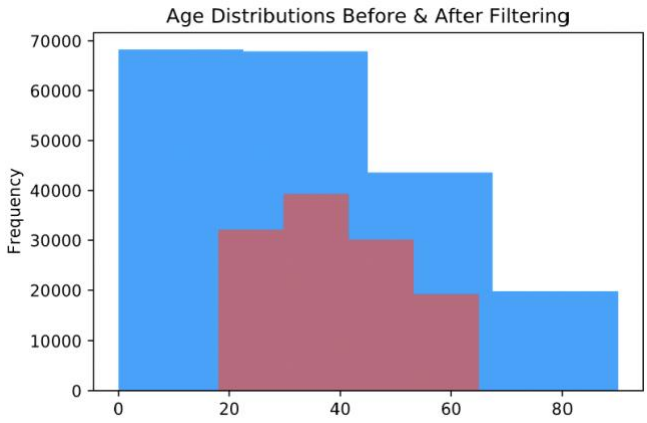
Our original data did not contain any NA values; instead, we dealt with "Not in universe" values, which acted as NAs. We choose to encode these specific cases as the majority class but may drop or impute them in the future based on kNN. With enough vectors, we may be able to avoid these altogether in the case of LVQ.

- While analyzing the data, we discovered that the highest percentage of Asians earned more than $50,000 yearly, with Whites following closely behind.
- Capital gain was a good predictor of wealth, creating a clear distinction between individuals who made $50,000 or more and those who did not.

- The biggest percentage of high-income persons were those who completed a PhD, Master's, or bachelor's degree.

Finally, marriage played an important impact in income, since married couples tend to have the highest proportion of high-income individuals, with men comprising most of the workforce.

After spending some time examining the data, we made various modifications. Listed below is an illustration of how we opted to bin the "age" variable.



In blue is the actual age distribution, whereas in red is a condensed version of our data. One of the ways we reduced our data was by focusing on respondents between the ages of 18 and 65, considering that these individuals would make up much of the full-time and part-time workforce. Following our exploratory data analysis and encoding, we arrived at reasonable datasets for our initial model iteration.

The dimensions of our manipulated train and exam data were as follows:
Training- (93562, 32)
Test- (49404, 32)

Final Label Imbalance:
We did not have a good 50-50 split of our labels, as is typical for real-world datasets. Rather, the ratio for our dependent variables was **88.37 % - 11.63 %** for **0-1** respectively, after filtering.

Models:
We had opted Naïve Bayes as our Base Model. It's the only generative model we use in our project considering our discrete binary labels and reasonably large data set.

We also opted to build rest our classifications models to compete with one other. Specifically comparing

- Logistic Regression and SVM
    - We aim to observe which model among these two will be perform better although they are both are apt for similar classification
    - Reason: We know that SVM and Logistic Regression are quite similar. SVM tries to finds the "best" margin distance between the hyperplane and the support vectors) that separates the classes thus reducing the risk of error on the data, while logistic regression does not, instead it can have different decision boundaries with different weights that are near the optimal point

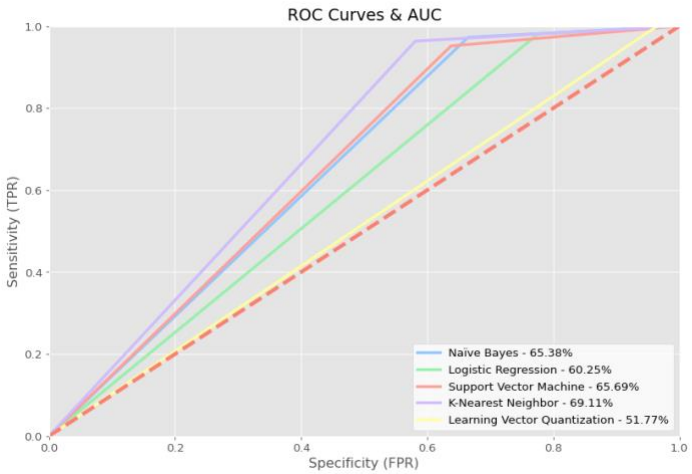
- K nearest neighbors and Learning Vector Quantization
    - At first glance one might think LVQ and kNN are essentially the same algorithm but the main distinction between the two is that LVQ makes a much more 'daring' choice, by attempting to synthesize a dataset into a number of 'sufficient' codebook vectors whereas kNN needs to memorize a whole dataset in order to make predictions on new data, LVQ
    - We used Euclidean distance as our main measure of similarity both the cases.
    - Here we wanted to see whether LVQ's condensed approach is more effective or not?

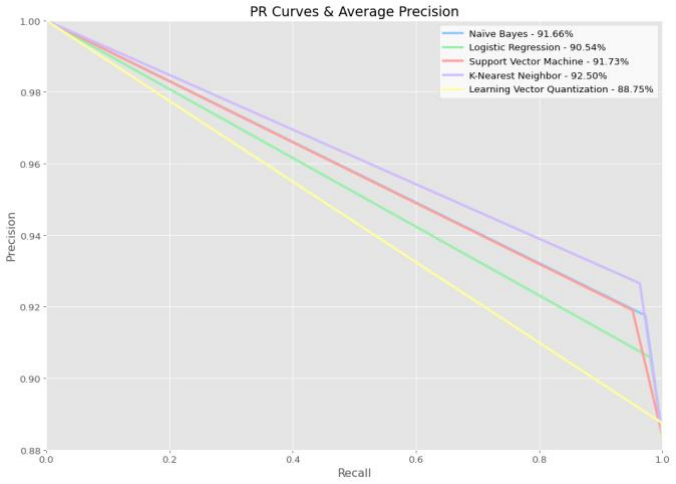
Initial Models – Without Fine Tuning and Data Normalization: (Accuracy on Test Data)

Without Normalization and No Fine Tuning of Parameters. Modelling Results:

Model	Accuracy	AUC
Naïve Bayes	89.8%	65.38%
Logistic Regression	89.2%	60.52%
SVM - Linear	88.3%	65.69%
KNN K = 5	**90%**	**69.11%**
LVQ	88.8%	51.77%

ROC and PR Curves of the above models





Conclusion from the above results:

- The clear winner here is KNN model with K =5, High Accuracy and AUC Values.
- SVM – Linear Model beat Logistic Regression with higher AUC Value.
- LVQ Model has AUC 50% only. Not a good model without Fine Tuning and Normalization. Here KNN won.

These models high Accuracy, Precision Values but low AUC Values. Low AUC values indicates that our initial model cannot successfully distinguish between positive class and negative class. **Clearly label imbalance's impacting our models.**

In summary, having high accuracy and precision values but low AUC, indicates that our model is not able to correctly predict the positive class, and it is not making good predictions overall.

Revesting Models – With Fine Tuning and Data Normalization:

For now, we're not addressing the label imbalance.

With Data Normalization and Fine Tuning of Parameters. Modelling Results:

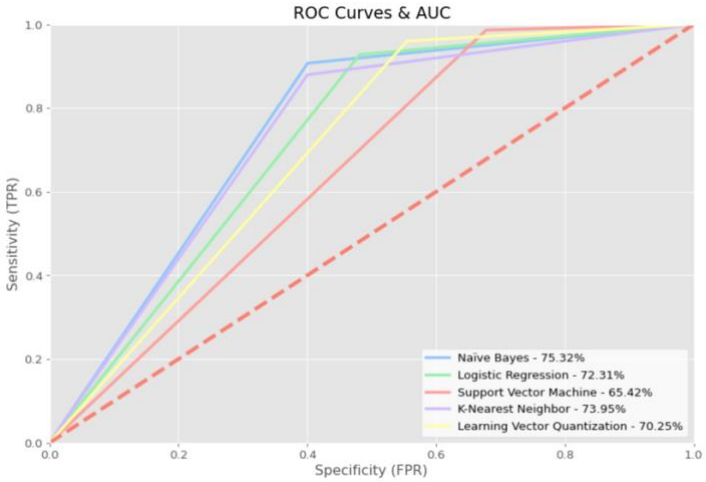
Model	Accuracy	AUC
Naïve Bayes	**87.1%**	**75.32%**
Logistic Regression (Lasso, Cross Validation)	88.3%	72.31%
SVM - RBF	90.9%	65.42%
KNN K = 2	84.7%	73.95%
LVQ	90%	70.25%

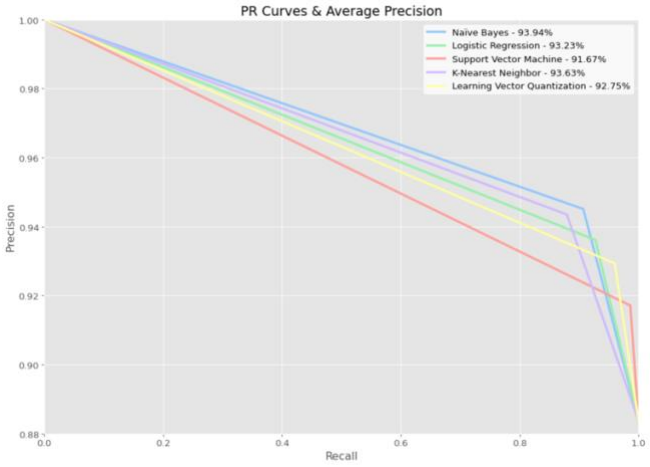
Why KNN =2?

K	Accuracy	AUC
2	**84.69%**	**73.95%**
3	89.63%	68.59%
4	88.8%	72.17%
5	90.2%	68.2%

As we already studied, Accuracy in imbalanced data set is not a good measure for choosing the model. K=2 has highest AUC value with comparable accuracy to all other K models. This is the reason we had choose K = 2 as the best parameter in KNN Models.

ROC and PR Curves of the above Fine Tune Models with Data Normalization





Conclusion from the above results | Fine Tuning and Data Normalization:

- Our AUC Values increased for different models (except SVM), with accuracies remaining nearly the same. Thus, Data Normalization using Standard Scalar, and fine tuning resulted in the performance increase of all the models (Except SVM).
- Comparing our Logistic Lasso and SVM RBF Finetuned models, Logistic Models have high AUC.
- Finetuning of KNN and LVQ with normalization had increased AUC Values. Now both models have comparable AUC values, in terms Accuracy LVQ wins here.
- Comparing all the models, Our Base Model Naïve Bayes has highest AUC.
    - Naïve Bayes having the highest overall AUC value, highest precision score and the lowest number of false positives, have the lowest accuracy scores.
    - These scores could be accounted due to the uniqueness of the data and trying to model the distribution yielded to be more effective rather than predicting from parameters by instance similarity.

Revesting Models – Dealing with Label Imbalance using SMOTE:

In our final data set dependent variables was **88.37 % - 11.63 %** for **0-1** respectively. Having a huge label imbalance.

Our AIM here is to deal with Label Imbalance using SMOTE to balance the labels, and generate the models again, understand the impact on Accuracy and AUC.
We didn't fine tune the models here.

SMOTE:
SMOTE, which stands for Synthetic Minority Over-sampling Technique, is a technique for oversampling a dataset to balance the class distribution. This is crucial since many machine learning techniques assume a balanced dataset and perform poorly on imbalanced datasets.

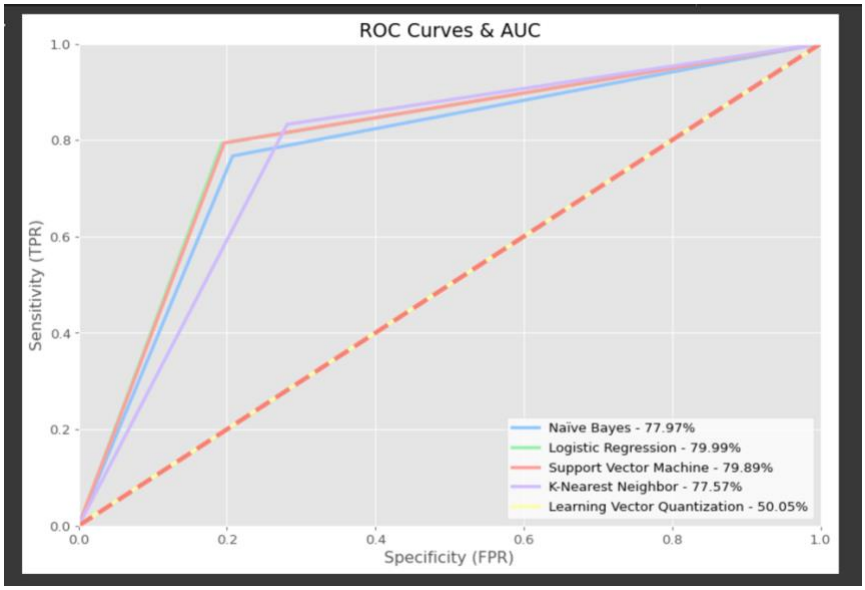
The minority class (i.e., the class with fewer instances) is oversampled in SMOTE by generating synthetic examples that are similar to the minority class's existing examples. This is accomplished by randomly selecting an instance from the minority class and identifying its k nearest neighbors. Synthetic instances are then developed by interpolating between the selected example and its neighbors.

This procedure is repeated until the minority class has as many instances as the majority class. The balanced dataset can then be used to build a machine learning model that performs better than it would on an imbalanced dataset.

Overall, SMOTE is a valuable machine learning technique for dealing with imbalanced datasets. It can assist in enhancing the performance of your model and making it more accurate when predicting fresh data.

Results | Revesting Models – With SMOTE:

Model	Accuracy	AUC
Naïve Bayes	78%	77.97%
Logistic Regression	**80%**	**79.99%**
SVM - RBF	**79.9%**	**79.89%**
KNN K = 5	77.6%	77.57%
LVQ	50.1%	50.5%



Conclusion from the above results | with SMOTE:

- Right now, for us it is difficult conclude, why by the use of SMOTE Resulted in higher AUC's for all our models but lower accuracy values. In general, the AUC, or area under the curve, is a measure of how well a model can distinguish between two classes, so a higher AUC value is generally better. However, accuracy is a measure of how well a model can predict the correct class for a given sample, so a lower accuracy value may indicate that the model is making more errors.
- Here we learnt how label imbalance can drastically impact ML Models power to distinguish between two classes.
- Logistic Regression and SVM Linear Models show nearly same performance, beating our baseline Naïve Bayes Model.
- LVQ needs to be fine-tuned here. We went the OG parameters for our initial study with SMOTE.

Need to do Further Investigations in SMOTE Models:

- We believe there is further room improvement coming to our SMOTE Models. Our plan of action:
    - Fine Tune SVM and Logistic Regression Parameters with Grid Search to obtain best parameters for these models.
    - Fine Tune KNN and LVQ Parameters by accuracy, understand how AUC Varies, select the best from them.

Final Outcomes Observed:

Finetuning and Normalization:

- While all our models post tuning saw an increase in their AUC, LVQ's performance increased by 20% after tuning.
- With all our tuned models we observed that we could predict earnings with a +90% precision.
- Although we spent long hours intensively experimenting and fine-tuning our various models, we were not able to beat our baseline model, Naïve Bayes. Thus, highlighting its accuracy and makings us re-evaluate our approach for this problem setting.

Our Final Model Evaluation metrics obtained:

	Not Normalized		Normalized and Finetuning		SMOTE	
Model	**Accuracy**	**AUC**	**Accuracy**	**AUC**	**Accuracy**	**AUC**
Naïve Bayes	89.80%	65.38%	**87.10%**	**75.32%**	78%	77.97%
Logistic Regression	89.20%	60.52%	88.30%	72.31%	**80%**	**79.99%**
SVM	88.30%	65.69%	90.90%	65.42%	**79.90%**	**79.89%**
KNN K = 5	**90%**	**69.11%**	84.70%	73.95%	77.60%	77.57%
LVQ	88.80%	51.77%	90%	70.25%	50.10%	50.50%

Future Study:

- As already mentioned, fine tuning our SMOTE Models and learn how accuracy and AUC values impacted.
- We believe that our exploratory study may be rethought, and that deleting or altering key factors could be advantageous.
- Using kNN to impute these missing variables might provide models with more accurate training data.
- Applying an unsupervised learning approach to the Census Dataset would permit the identification of sub-groups, followed by training and testing of models based on the sub-groups, as opposed to training/testing the complete dataset.
- We might potentially add other datasets to the Census Income dataset in order to get better results that can forecast income by region, as this would allow distinction between regions and increase the model's precision and accuracy.