Exploring Machine Learning Models on Philippine Labor Force Survey

L. A. Apetreor[1] And A.T. Clavano[2] And Y. Homssi[3]
[1]*Department of Software Technology, college of computer science, de la salle university, taft avenue, manila, philippines*
STINTSY, S17 and Group 5
date performed: 31/03/2025

ABSTRACT

This research paper delves into the applications of different machine learning models to predict a person's occupation based on a Philippine Labor Force Survey with data collected in April 2016. The group used machine learning algorithms such as K-Nearest Neighbors (KNN), Naive Bayes- including both Gaussian Naive Bayes (GNB) as well as Multinomial Naive Bayes (MNB), and Neural Networks to conduct this study. The study primarily examines the effectiveness and accuracy of the different models when it comes to predicting the occupation of a respondent given all other relevant features.

INTRODUCTION

This paper contains the processes and results of our project for STINTSY in Term 2 AY 2024-2025. The group was given 2 choices of datasets and chose the Labor Force Survey 2016. This dataset focuses on employment, labor force participation, and job-related details. It includes information about individuals' jobs, hours worked, job search activities, and employment status. It contains around 50 features, including employment status, occupation, hours worked, and job search methods. The group trained 3 different kinds of models: K-Nearest Neighbors, Naive Bayes, and Neural Networks. Given the dataset, the group set our target variable, i.e. what we are trying to predict, to the Primary Occupation variable (PUFC14_PROCC). Therefore, the task at hand is to train machine learning models to predict the respondent's primary occupation given the remaining features.

METHODOLOGY

The first part of the project required our group to conduct Data Cleaning. There are three different kinds of data in our dataset: Binary-Categorical, Categorical, and Numerical which are grouped accordingly due to the way they are dealt with during cleaning. Before proceeding with data cleaning, the group dropped variables that would not be important to our task. (date variables, ID numbers, all variables related to unemployment, and NEWEMPSTAT referring to employment status) For all variables, the group checked and converted the data to the correct data type. (i.e. Binary-Categorical data types were converted from object to string, then to 'yes' or 'no') The next step required handling of null/blank variables. Categorical variables used mode imputation, Numerical variables used median imputation, and Binary-Categoricals simply imputed the "No" option on blank or whitespace classes, which according to the valueset mean "not applicable". At the end of the Data Cleaning process, a .csv file was created containing the cleaned dataset.

Exploratory Data Analysis (EDA):

The EDA statistic the group has decided to use to describe the relationship between the numerical features and the target variable (categorical) were

P-statistics and F-statistics from ANOVA. This is because the group wanted to find the correlation between the variance of each numerical feature and the classes of the target variable to see how the features correlated via the F-statistic. The group used the P-statistic to figure out if there is any statistical significance to the findings inferred from the F-statistic.

The EDA for Categorical Variables vs Categorical Target Variable explored both univariate and bivariate methods of analysis. It also used YData-Profiling as a guide. For the univariate analysis, the frequency count of each possible value for each categorical variable was evaluated. The bivariate analysis used Cross-Tabulation to create a frequency table of each categorical variable compared to the target variable. As a continuation to the bivariate analysis, Cramer's V was used to measure the strength of association between the variables. A contingency table was also created to support Cramer's V findings.

K-Nearest Neighbors (kNN):
This model was selected because it's a fairly simple model to implement compared to the other models. The group wanted to see how a simpler model could fare against more complex models. kNN makes predictions based on a data point's similarity to its neighboring data points. It contains two hyperparameters: "k" neighbors to consider and a "distance metric."

A copy of the cleaned dataset is loaded into a DataFrame object. Since kNN requires the processed data to be numerical, the group used scikit-learn's LabelEncoder() to convert the binary-categorical and categorical data into numerical format. This is done for all columns except PUFC14_PROCC. From there, the dataset is split into training (80%) and test sets (20%) via train_test_split().

The initial model training used sklearn's KNeighborsClassifier() with the following parameters: n_neighbors=5, algorithm=kd_tree, and metric=euclidean. These values were picked arbitrarily except the metric, which was the default. This resulted in an accuracy score of 61.003%

The Error Analysis for kNN began by evaluating the classification report of the initial model. The classification report showed Armed Forces Occupations and Elementary Occupations are 2 classes that the model had a hard time classifying, due to their low Precision, Recall, and F1-Score. A confusion matrix showed that the top 5 misclassified classes are Elementary Occupations, Service and Sales Workers, Skilled Agricultural, Forestry, and Fishery Workers, Craft and Related Trade Workers, and Professionals. Finally, setting a threshold of 0.5 showed that 'Armed Forces Occupations', 'Clerical Support Workers', 'Craft and Related Trades Workers', 'Elementary Occupations', 'Managers', 'Plant and Machine Operators and Assemblers', 'Professionals', 'Service and Sales Workers', 'Skilled Agricultural, Forestry and Fishery Workers', 'Technicians and Associate Professionals' are imbalanced.

Section 8 of the notebook was on Improving Model Performance. The group started by standardizing the data using three different scalers: StandardScaler(), RobustScaler(), and MinMaxScaler(). Standard Scaler standardizes features by removing the mean and scaling to unit variance. The group noticed that there were many outliers, and tried RobustScaler(). This resulted in negative values, which does not fit for a real-world dataset. Finally, a third scaled version of the dataset was created using MinMaxScaler() to compensate for negative values.

Feature selection was conducted on each subset next using SelectKBest(), which removes all but the highest scoring features. For this method, hyperparameter k was set to 10 top features, and the score_func was set to f_classif, following the default.

All three sub-datasets were put through Principal Component Analysis (PCA) iteratively. PCA was done to reduce dimensionality in the dataset. A threshold of 0.95 (95%) was considered when finding the number of components, meaning the PCA process is to find the best number of components to consider to keep 95% of the variance. Then, using the results from the initial

PCA, a Pipeline was created with GridSearchCV. GridSearch tests out different hyperparameters and finds the best ones. This is done for all subsets. A final evaluation between the scaled sets and the original split data was conducted. This is done 5 times and averaged.

Naive Bayes (NB):

Both Gaussian Naive Bayes (GNB) and Multinomial Naive Bayes (MNB) were selected as they are both computationally simple models that both assume feature independence, allowing these models to be simplified and work efficiently. These models were primarily used as a starting point for our classification problem, while also using a real-world dataset and being computationally efficient due to its simplistic nature.

The dataset has also been shown to have a high dimensionality, leading the group to decide that NB models may be efficient as they are not very prone to overfitting due to its use of probabilistic modeling.

MNB was seen by the group as well suited for handling the categorical data in this dataset in general, as it allows data to be represented in frequency counts easily incorporating the likelihood of certain categorical variables which the dataset is mostly composed of.

GNB was also seen as another option for handling the dataset as it is also a simple probabilistic model that is well suited for predicting continuous features which the dataset also has a large proportion of. This model was primarily considered due to its ability to take into account class imbalance by taking into account prior probabilities for each class which are directly proportional to the frequency of said class in the dataset, leading us to assume that it could make accurate predictions even when some occupations are underrepresented in the dataset while retaining a simpler computational complexity compared to other models.

In section 6 of the notebook, the group used a Label Encoder in order to encode the categorical data into numerical data which allowed the use of both GNB and MNB models. The group trained the GNB and MNB models without any optimizations to serve as a baseline model reference to compare its statistics with any optimizations. We shuffle and run each model 5 times and find the average training and testing accuracies for those 5 iterations. The hyperparameters of GNB and MNB by default are respectively a var_smoothing of 1e-09, and an alpha value of 1.0. The var_smoothing hyperparameter of GNB is used to add a small value to each feature's variance in order to prevent division by zero while calculating Gaussian likelihood while the alpha hyperparameter for MNB is a smoothing parameter that is used to ensure non-zero probabilities for all features. All NB models have the hyperparameter of fit_priors in common which is a boolean flag wherein if set to true, allows the model to estimate priors from the training data, and while false allows the model to assume uniform priors.

In section 7 of the notebook, the group first checked for class counts and class priors, which are observed to be imbalanced. The group has concluded that optimizations such as SMOTE for oversampling, scalers such as Normalizer and MaxAbsScaler, feature selection algorithms such as Select-K-Best , and/or PCA are vital in improving model performance.

In section 8 of the notebook, the group has implemented said optimizations, comparing each optimization made to the baseline model. The group ended up utilizing an iterative approach to find the best combination of optimizations, iterating over hyperparameter K for select-k-best feature selection, iterating over alpha and var_smoothing hyperparameters for MNB and GNB respectively, iterating over different n_components hyperparameter to find the best n_component value for PCA, and iterating over the 2 different scalers Normalizer and MaxAbsScaler.

Note that some optimizations are incompatible for either MNB and GNB, such as PCA, adjusted priors, and sample weights being incompatible with MNB as MNB is incompatible with negative counts.

Neural Networks (NN):

To optimize the neural network, hyperparameter tuning was performed using both Random Search and Grid Search, ensuring that the model configuration was well-suited to the classification task. The hyperparameters tuned included the number of hidden layers, number of neurons per layer, learning rate, dropout rate, and optimizer selection. The search space for each hyperparameter was carefully chosen to balance computational efficiency and model performance.

A 5-fold cross-validation strategy was implemented to ensure that the model generalized well to unseen data, reducing variance and mitigating overfitting. The network architecture comprised an input layer, two hidden layers with ReLU activation, and an output layer utilizing Softmax activation for multi-class classification. The model was trained using the Adam optimizer with sparse categorical cross-entropy as the loss function, which is suitable for classification tasks with integer-labeled targets.

L1 regularization was introduced to allow feature selection by penalizing unnecessary weights. However, a comparative analysis between models trained with and without L1 regularization revealed that while L1 helped in feature reduction, it negatively impacted accuracy. This trade-off was analyzed by evaluating feature importance, model accuracy, and loss values before and after regularization.

RESULTS

- (Arial 11, justified, no indentation)
- Presents pertinent results of the experiment
- Organizes data into tables, figures, graphs, etc. (whichever is appropriate)
- Include all tables, plots, and sample (only for 1 trial) calculations
- Should provide labels with short caption for each tables, figures, graphs, etc.
- Label for *tables* should be on top of the table while label for *figures* should be below the figure

K-Nearest Neighbors Results:

The results of the final evaluation showed the following:

Table 1. kNN Model Comparison

Model	**Scaling Method**	**K-neighbors**	**Algorithm**	**Metric**	**Average Test Accuracy**
Initial	None	5	kd_tree	Euclidean	61.0 %
stdscaler	StandardScaler()	9	ball_tree	Manhattan	71.5 %
rbst	RobustScaler()	9	auto	Manhattan	59.8 %
minmax	MinMaxScaler()	9	auto	Manhattan	73.6 %

The best performing kNN model was the one that was standardized using StandardScaler() (71.5% average accuracy) with 9 neighbors, using the ball_tree algorithm, and using a Manhattan distance metric. The worst performing kNN model was the one standardized using RobustScaler() (59.8% average accuracy) with 9 neighbors, automatic algorithm, and Manhattan distance metric.

Naive Bayes Results:

The results of the different optimizations have resulted in the following:

Table 2. NB Model Comparison

Model	**Average Test Accuracy**
GNB (baseline)	39.97% ± 0.51%
MNB (baseline)	42.63% ± 0.22%
GNB + UNIFORM PRIORS	38.85% ± 0.56%
GNB + SMOTE	39.60% ± 0.54%
MNB + SMOTE	40.99% ± 0.0025%
MNB + MAXABS SCALER	42.60% ± 0.26%
MNB + NORMALIZER	30.65% ± 0.01%

GNB + MAXABS SCALER	38.95 ± 0.52%
GNB + NORMALIZER	42.44% ± 0.47%
GNB + PCA	52.20% ± 0.37%
BEST MODEL (ITERATIVE)	52.16% ± 0.42%

The best performing model was GNB with PCA as the most significant optimization implemented, being roughly the same accuracy as the best model that was iterated over to find the best pair of optimization strategies. The best NB model derived from iteration is a GNB using a Max-Abs Scaler, with a K value of 17±1 for select-K-best feature selection, a var_smoothing value of 1e-08±1e-01, and a n_compontents value of 12±1 for PCA, as it varies per shuffle. It can be assumed that it would have been more computationally efficient while retaining the same accuracy to just have implemented PCA, and to iterate over its hyperparameter to find the best n_compontents value.

DISCUSSION

Our group worked with a large dataset, which most likely had complex patterns. Out of all of the models, the Neural Network model outperformed the kNN and Naive Bayes models in terms of accuracy. It's notable that kNN, a far simpler model, came in second place. This could indicate the dataset's features were fit for distance-based comparison, and not for the feature independence assumption within Naive Bayes.

Although the Neural Network model performed the best, it was also the most complex and computationally expensive. It also took way more time to process compared to the other two models. This shows the importance of balancing the trade-off between performance and computational costs considering the size and dimensionality of the dataset.

CONCLUSION AND RECOMMENDATION

Aside from trying different variants of the models that were trained and evaluated in this paper, the group suggests exploring more feature transformations or selection methods to improve the individual model performances, as other methods may be more suitable for the dataset. Finally, there is potential for combining the models to create an ensemble model, leveraging the strengths of each model to improve overall performance.

For feature selection in kNN, there was an initial method that was taken out in favor of using SelectKBest. The inital method used the findings from the EDA to select the features to exclude and include, as shown below:

```
numerical_vars = ['PUFC25_PBASIC', 'PUFC05_AGE', 'PUFC18_PNWHRS', 'PUFC19_PHOURS', 'PUFC28_THOURS'] # based on anova results
bincateg_vars = ['PUFC08_CURSCH', 'PUFC09_GRADTECH', 'PUFC20_PWMORE', 'PUFC22_PFWRK'] # by relevance
categ_vars = ['PUFC43_QKB', 'PUFC16_PKB', 'PUFC04_SEX', 'PUFC23_PCLASS', 'PUFURB2K10'] # by highest cramers v
```

The group suggests using this method of manual selection to see if it improves the model performance in future experiments to come.

REFERENCES

6.3. Preprocessing data. (n.d.). scikit-learn. Retrieved March 31, 2025, from https://scikit-learn.org/stable/modules/preprocessing.html#scaling-features-to-a-range

Carrascosa, I. P. (2025, January 7). A Complete Guide to Cross-Validation. https://www.statology.org/complete-guide-cross-validation/

D, P. (2024, September 17). Optimizing k-nearest neighbors for large datasets in scikit-learn. peerdh.com. https://peerdh.com/blogs/programming-insights/optimizing-k-nearest-neighbors-for-large-datasets-in-scikit-learn

GeeksforGeeks. (2024, November 16). How to reduce KNN computation time using KD-tree or ball tree? https://www.geeksforgeeks.org/how-to-reduce-knn-

computation-time-using-kd-tree-or-ball-tree/

Gokte, S. A. (n.d.). Most popular distance metrics used in KNN and when to use them. KDnuggets. Retrieved March 31, 2025, from https://www.kdnuggets.com/2020/11/most-popular-distance-metrics-knn.html

LaViale, T. (2023, March 3). Deep dive on KNN: Understanding and implementing the k-nearest neighbors algorithm. Arize AI. https://arize.com/blog-course/knn-algorithm-k-nearest-neighbor/

Sklearn.feature_selection.SelectKBest. (n.d.). scikit-learn. Retrieved March 31, 2025, from https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html

Sklearn.model_selection.GridSearchCV. (n.d.). scikit-learn. Retrieved March 31, 2025, from https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

Venkatachalam. (2019, March 9). Should I normalize or standardize my dataset for knn? Stack Overflow. https://stackoverflow.com/questions/55073423/should-i-normalize-or-standardize-my-dataset-for-knn,

APPENDIX

- https://psada.psa.gov.ph/catalog/67

CONTRIBUTIONS

Member	**Contribution(s)**
Apetreor, Lee Jacob Marcus A.	• Gaussian Naive Bayes • Multinomial Naive Bayes • Data cleaning for Numerical variables • Data cleaning for Binary-Categorical variables • ANOVA EDA for numerical variable and target variable • Numerical variable EDA: correlation heatmap matrix
Clavano, Angelica Therese	• kNN Model • Introduction • Data cleaning for Categorical Variables • Univariate Analysis, Bivariate Analysis EDA for Categorical Variables
Homssi, Yazan M.	• Neural Network • Data cleaning for Categorical Variables • Cramer's V, and countplot frequency distribution EDA for Binary-Categorical Variables • Bivariate Analysis EDA for Categorical Variables