# *Using Bayesian Networks and SVM to Determine Income*

Group Members: David Cordero and Daniela Johnson

## I. INTRODUCTION

There are many different factors that come into play when determining an adult's income. In this project, we will use two classification models, the Bayesian Network and the Support Vector Machine (SVM), to predict whether an adult will make $50,000 or less a year or over $50,000 a year. Our predictive features are age, work class, final weight, education, education number, marital status, occupation, relationship, race, sex, capital gain, capital loss, hours worked per week, and native country.

To begin, we had to preprocess the data. We noticed there were missing values in the work class column (2799 values), the occupation column (2809 values), and the native country column (857 values). We filled in the missing values with each feature's mode. Since SVM only works with numerical data, we used one-hot encoding to make all categorical features numerical. We knew one-hot encoding would suffice since the features we converted were nominal; we converted our target variable, income, to 0 if someone made $50,000 or less and 1 if someone made over $50,000. We deleted the final weight feature because these weights were added after data collection, and we deleted the education feature because the education number feature already represents it. Finally, we used SMOTE to balance our data to represent both incomes equally (over $50K and $50K or less). Incomes that were over $50,000 were heavily undersampled.

Our analysis revealed key insights into the performance of Bayesian Networks and Support Vector Machines in predicting income levels. We found that Bayesian Networks were not the most accurate in predicting income based on the features and that modifying the training and testing data size did not help much either. Meanwhile, for the SVM, the choice of kernel function and hyperparameters played a pivotal role. The cubic kernel showed improved accuracy for complex data distributions, but it was more prone to overfitting compared to the linear kernel. By carefully tuning the regularization parameter C, we were able to balance bias and variance effectively, resulting in a robust classification model. These findings underline

the importance of hyperparameter optimization in machine learning models for income prediction.

## II. METHODS

### *a) Bayesian Network*

To define and train the Bayesian Network, we used a feature selection method to pick the most relevant features, which were then used to train the model. The method that we chose is based on structure learning. In this approach, it is important to create a network structure. HillClimbSearch finds dependencies between variables within the data and then selects the ones with the best Bayesian Information Criterion (BIC) score. These tools infer dependencies between variables based on statistical measures, such as conditional independence tests or scoring functions. The features that are chosen in this method are then used to estimate the conditional probability distributions. Using relevant features can reduce the risk of overfitting. The model is then fit to the training data and predicts classes on the testing data. To evaluate the performance of the model, we implemented classification reports to calculate accuracy, precision, recall, and F1-score.

We experimented with different test sizes as a hyperparameter. We ran the Bayesian Network model multiple times using 20% of the dataset as the testing set, 25%, 35%, and 40%. When the testing data was 20% of our total data, our training data was the remaining 80%. We ran a classification report and found the predictions' accuracy for each test size to compare which size obtained the best results.

### *b) SVM*

For this project, we used an SVM model, implemented with the CVXOPT optimization library, and tested it with a custom polynomial kernel. We started by cleaning and preparing our dataset by replacing missing values, converting categorical values to numerical codes, removing duplicate rows, and dropping columns with low variance, as they do not contribute to the model. To ensure consistent scaling across the features, we also normalized the data. SMOTE played a critical role in balancing the dataset, allowing the models to learn from a more diverse set of samples. Without SMOTE, the models would have

been biased toward the majority class, leading to poor generalization. This preprocessing step significantly enhanced the models' ability to classify minority class instances, as reflected in the improved recall scores.

The SVM was set up with a polynomial kernel to map the data into a higher-dimensional space. The kernel function we used was $K(x_1, x_2) = (1+x_1*x_2)^{degree}$, allowing us to experiment with different levels of nonlinearity by adjusting the degree parameter. We solved the SVM optimization problem using quadratic programming, adding a small regularization term to the kernel matrix for numerical stability. Hyperparameters also played a large role in refining the model. The regularization parameter C, which controls the trade-off between making the margin wider and allowing some classification errors, was set to 1 to strike a balance between over- and underfitting. We also tested different polynomial degrees to see how nonlinearity affected the results. To evaluate the model, we used 5-fold cross-validation, which let us test the SVM on multiple data splits for a more reliable assessment.

The model calculated support vectors, weights, and biases to define the decision boundary. Predictions were made using sign(w*x+b). For each cross-validation fold, we measured the performance's accuracy. This helped us understand how well the model was doing and guided us in tweaking the hyperparameters and kernel settings to improve its performance.

## III. RESULTS

### *a) Bayesian Network*

The first step in implementing the Bayesian model was to implement the feature selection method. To begin, we had to find the graph structure of the network. The results of the relationships between the different variables from the dataset are below in Table 1.

| Independent Variable | Dependent Variable |
|---|---|
| Capital gain | Income |
| Income | Marital status: married-spouse |

| | |
|---|---|
| Income | Relationship: own-child |
| Income | Marital status: Never-married |
| Income | Capital loss |
| Income | Occupation: Priv-house-serv |
| Income | Relationship: other-relative |
| Income | Native-country: Columbia |
| Income | Native-country: Guatemala |
| Income | Work class: never-worked |

Table 1: Model Nodes Found using HillClimbSearch and BIC score

In Table 1 the first variable listed on the left has an influence on the second variable listed on the right–or that the second variable is dependent on the first variable. Looking at the first example, capital gain has an impact on income. This could imply that the amount of capital gain a person has influences their income classification, and in this case, that would mean whether they make more or less than $50,000 a year. Looking at the second example, income level influences marital status. Income seems to influence the other variables the most, which makes sense as it is the target variable.

To train the model we only used the most relevant features. The model nodes are income, marital status: Married-spouse, relationship: Own-child, marital status: Never-married, capital loss, occupation: Priv-house-serv, relationship: Other-relative, native-country: Columbia, native-country: Guatemala, work class: Never-worked. An error we encountered while training the data was due to the column names differing after using One-hot encoding. The model nodes did not match our original feature names because our model nodes were more specific, for example not just stating native country, but which native country. To fix this error, we found the intersection of the model's node names and the column names in X_test:

X_test_filtered = X_test.loc[:,list(model_nodes.intersection(X_test.columns))].

Then, we proceeded to run the Bayesian Network model using only our relevant features. We experimented with different test sizes to see if we could obtain better results. Our results are shown in Table 2 below. Our accuracy did not improve much, but normally, reducing the size of the testing data tends to yield better results because a larger training set improves the model's ability to generalize to unseen data. A larger training set is less likely to overfit to specific training samples. On the other hand, a larger test set provides a more reliable evaluation of the model, but a smaller training set can lead to the model not learning underlying patterns within the data.

Since our accuracy exhibits negligible change as the test size increases, the model has learned the patterns in the training data well and is generalizing to unseen data even at the largest test size of 40%, meaning 60% of the data was used to train the model. 60% of the data is sufficient for this model because this dataset is very representative of the overall population distribution.

| Test Size | Accuracy |
|---|---|
| 0.2 | 71.08% |
| 0.25 | 71.23% |
| 0.3 | 71.42% |
| 0.35 | 71.53% |
| 0.4 | 71.53% |

*Table 2: Bayesian Network Model's Accuracy of Varying Test Sizes*

We conducted a classification report for each train and test size as well. All 5 sizes had the same exact precision. 93% of the people who were predicted to make under $50K or less actually made it, and 45% of predictions of who would make over $50K were correct. The recalls of all train and test sizes only ranges by 1%. Approximately 67% to 68% of those who made an income of $50,000 or less were identified correctly and approximately 83% to 84% of those who made an income over $50,000 were identified correctly. Our classification report

also gave us the F1-score which calculates the harmonic mean between precision and recall values. Again all train and test sizes have very similar percentages. Our lower income classification has a higher percentage of 78% and our higher income classification has a lower percentage of 58%. This means that our model's predictions are often more correct for those who make an income of $50K or less than those who make an income over $50K. Overall, after experimenting with hyperparameters and analyzing accuracy, precision, recall, and F1-score, the Bayesian Network model is not a great model to predict income for this dataset.

*b) SVM*

| | pcost | dcost | gap | pres | dres | Degree 3 | C .1 |
|---|---|---|---|---|---|---|---|
| 0: | -3.4155e+02 | -1.5436e+02 | 3e+03 | 2e+01 | 3e-16 | | |
| 1: | -5.2789e+01 | -1.4580e+02 | 1e+02 | 2e-01 | 4e-16 | | |
| 2: | -5.7232e+01 | -6.4208e+01 | 8e+00 | 8e-03 | 3e-16 | | |
| 3: | -6.2051e+01 | -6.2127e+01 | 8e-02 | 9e-05 | 3e-16 | | |
| 4: | -6.2100e+01 | -6.2100e+01 | 8e-04 | 9e-07 | 3e-16 | | |
| 5: | -6.2100e+01 | -6.2100e+01 | 8e-06 | 9e-09 | 2e-16 | | |
| Optimal solution found. | | | | | | | |

| | pcost | dcost | gap | pres | dres | Degree 3 | C 1 |
|---|---|---|---|---|---|---|---|
| 0: | -3.4155e+02 | -1.5436e+02 | 3e+03 | 2e+01 | 1e-16 | | |
| 1: | -5.2789e+01 | -1.4580e+02 | 1e+02 | 2e-01 | 4e-16 | | |
| 2: | -5.7232e+01 | -6.4208e+01 | 8e+00 | 8e-03 | 3e-16 | | |
| 3: | -6.2051e+01 | -6.2127e+01 | 8e-02 | 9e-05 | 3e-16 | | |
| 4: | -6.2100e+01 | -6.2100e+01 | 8e-04 | 9e-07 | 3e-16 | | |
| 5: | -6.2100e+01 | -6.2100e+01 | 8e-06 | 9e-09 | 3e-16 | | |
| Optimal solution found. | | | | | | | |

| | pcost | dcost | gap | pres | dres | Degree 1 | C .1 |
|---|---|---|---|---|---|---|---|
| 0: | -3.4155e+02 | -1.5436e+02 | 3e+03 | 2e+01 | 1e-16 | | |
| 1: | -5.2789e+01 | -1.4580e+02 | 1e+02 | 2e-01 | 4e-16 | | |
| 2: | -5.7232e+01 | -6.4208e+01 | 8e+00 | 8e-03 | 3e-16 | | |
| 3: | -6.2051e+01 | -6.2127e+01 | 8e-02 | 9e-05 | 3e-16 | | |
| 4: | -6.2100e+01 | -6.2100e+01 | 8e-04 | 9e-07 | 3e-16 | | |
| 5: | -6.2100e+01 | -6.2100e+01 | 8e-06 | 9e-09 | 3e-16 | | |
| Optimal solution found. | | | | | | | |

| | pcost | dcost | gap | pres | dres | Degree 1 | C 1 |
|---|---|---|---|---|---|---|---|
| 0: | -6.2100e+02 | -1.4500e+03 | 3e+03 | 1e+00 | 2e-16 | | |
| 1: | -5.1625e+02 | -8.9787e+02 | 4e+02 | 1e-02 | 5e-16 | | |
| 2: | -6.0982e+02 | -6.2554e+02 | 2e+01 | 6e-04 | 4e-16 | | |
| 3: | -6.2089e+02 | -6.2105e+02 | 2e-01 | 6e-06 | 5e-16 | | |
| 4: | -6.2100e+02 | -6.2100e+02 | 2e-03 | 6e-08 | 2e-16 | | |
| 5: | -6.2100e+02 | -6.2100e+02 | 2e-05 | 6e-10 | 3e-16 | | |
| Optimal solution found. | | | | | | | |

*Table 3: SVM Metrics with Varying Degrees and Hyperparameters*

In this model, we evaluated the performance of the SVM using mainly two different kernel functions (linear and cubic) while varying the regularization parameter C. The goal was to assess how the choice of kernel and hyperparameters influenced the model's classification performance. Below, we discuss the effects of different learning methods, modifications to traditional SVM methods, and variations in hyperparameter settings.

When comparing the linear and cubic kernels, we observed notable differences in model performance. The linear kernel is typically more efficient and works well when the data is linearly separable or nearly linear. In this case, it provided faster computation times and generally performed well, especially at lower values of C. The cubic kernel, on the other hand, is capable of capturing more complex relationships between the data points. For higher values of C, the cubic kernel exhibited a higher sensitivity to the training data, leading to potential overfitting, but it also demonstrated improved accuracy on datasets with complex

decision boundaries. The parameter C mainly controls the trade-off between maximizing the margin and minimizing the classification error. With small values of C, both the linear and cubic kernels had more regularized models, which often resulted in higher bias but lower variance. This reduced the risk of overfitting, especially with the cubic kernel. On the other hand, larger values of C allowed the model to fit the training data more closely, reducing bias but increasing the risk of overfitting, particularly for the cubic kernel. The linear kernel was generally less sensitive to changes in C, as its simpler structure did not exhibit as much variance with large values of C.

The choice of C was particularly important in determining model performance. With low values of C, the SVM tended to create broader margins, which led to underfitting, as the model was not complex enough to capture all the data patterns. For larger values of C, the model could fit the training data more closely, but this increased the likelihood of overfitting, especially for the cubic kernel, where the decision boundary became more sensitive to individual data points. As expected, the cubic kernel showed an increased ability to handle complex data distributions at higher values of C.

The choice of C was particularly important in determining model performance. With low values of C, the SVM tended to create broader margins, which led to underfitting, as the model was not complex enough to capture all the data patterns. For larger values of C, the model could fit the training data more closely, but this increased the likelihood of overfitting, especially for the cubic kernel, where the decision boundary became more sensitive to individual data points. As expected, the cubic kernel showed an increased ability to handle complex data distributions at higher values of C, though this came at the cost of overfitting. The linear kernel, however, was more stable across various C-values, maintaining a balance between bias and variance. In conclusion, the choices of kernel and hyperparameter settings significantly influence the performance of the SVM.

The metrics in Table 3 collectively monitor the progress of the optimization process. Pcost and dcost both represent the objective function value, but for the primal and dual problems, respectively. These values steadily approach one another. The primal problem is the original optimization problem that we are solving, whereas the dual problem is derived from the primal problem and provides a lower bound to the optimal primal objective. Gap is the

difference between pcost and dcost, measuring how far the current solutions to the primal and dual problems are from optimality.

Pres and dres are the primal and dual residuals, measuring the violation of constraints of their respective problems. These residuals indicate how far the current solution is from satisfying the constraints of our problem. Lower values here signify better feasibility. Convergence is achieved when the duality gap approaches zero, and the residuals are minimized, signifying that an optimal and feasible solution has been found. Each of the table's rows represents the 5-fold cross-validation that we performed. The model's accuracy was unrelated to the differences in hyperparameters and kernel degree, as it stayed constant at 75.5%.

*c) Practical Implications of Results*

The insights we have gotten from the income prediction model have significant applications in many other sectors. In economic and policy decision-making, these models can identify demographic groups most likely to experience low income based on factors like education level, age, and occupation. Targeted interventions, like educational funding, job training programs, or tax incentives, can then be designed to address these disparities.

Organizations seeking to improve diversity and equity in hiring can use insights from these models to adjust hiring strategies. The models reveal how particular features are correlated with income predictions, which might mirror biases in the labor market. Companies can analyze such correlations to ensure hiring practices are equitable and merit-based.

Banks, lending institutions, and insurers can leverage these models to make data-driven decisions. Income prediction is a critical component in evaluating loan eligibility. Accurate classification of income levels helps financial institutions assess repayment risk and determine interest rates. SVM models, particularly with cubic kernels, could help insurers predict future earning potential and adjust premium amounts accordingly, ensuring fairness and profitability.

This study also highlights broader implications for the responsible use of AI in income prediction. Disparities in classification accuracy across demographic groups could indicate potential biases in the dataset or model, which should be addressed to ensure equitable

outcomes. For example, SMOTE was necessary to correct class imbalance, but further steps might be needed for sensitive applications like hiring or credit scoring. By revealing the importance of specific features, Bayesian Networks can make income prediction more interpretable. This transparency is critical in applications where decisions impact people's lives, as it ensures accountability and public trust in AI systems.

## IV. CONCLUSION

This study explored the application of Bayesian Networks and Support Vector Machines to predict whether an adult's income exceeds $50,000 based on demographic and economic features. Key findings included the data's ability to generate unseen data no matter the training size through the Bayesian Network model and the impact of kernel choice and regularization in SVM.

For the Bayesian Network, the HillClimbSearch method identified relevant dependencies between features, achieving a highest classification accuracy of 71.53%. This method performed best when its training set was smallest and its testing set was largest, but not by much. This proved that the preprocessed dataset was very representative of the overall population. Techniques such as SMOTE and feature selection are very important in assisting the model to notice underlying patterns in the data, but not overfit them. The SVM model demonstrated that the cubic kernel could capture more complex decision boundaries, but it was more sensitive to overfitting at higher regularization values. In contrast, the linear kernel provided a more stable and computationally efficient option, especially for simpler data distributions. Our experiments underscore the importance of preprocessing steps, such as balancing datasets with SMOTE and transforming categorical variables, to enhance model performance. The results also highlight that the choice of kernel functions, and hyperparameters can influence a model's reliability. Future work could focus on incorporating ensemble methods, additional preprocessing techniques, or testing more kernel degrees and hyperparameter values.