### **Checkpoint 3 Report - CPSC 6300 Hubig**

### **Model Choice & Justification**

The response variable for our model to predict is the binary completion status of a task accepted by a crowdsource worker: 1 for a task that is completed, 0 for a task that was not completed. To predict this binary outcome, we must select a classification model. The first model chosen for classification was a K-Nearest Neighbors model, which is a relatively simple method where an observation is classified based on the category of close data points in the training set. For our second model, we decided to go with a support vector machine for classification, which attempts to classify data by splitting and spacing them along a decision boundary. We chose this model because it offered a more nuanced approach to classifying a dense dataset, and the high dimensionality of our inputs would not be a problem.

### **Model Analysis - Summary**

The Support Vector Machine (SVM) model includes different types of kernels (algorithms that determine the similarity between input vectors). We chose a ‘rbf’ or Radial Basis Function (RBF) kernel because the input data had more clustered points from our EDA and was non-linear. We left the hyperparameter ‘C’, which controls the trade-off between the insensitive and sensitive loss, at the default of 1.0 value. The smaller the ‘C’ means that the model will be more lenient in allowing larger errors and larger value means that the model will minimize the insensitive loss more. Figure 1 shows the score (mean accuracy), accuracy, precision, recall, and F1-Score (harmonic mean of precision and recall) of the SVM model, where the closer the value is to one, the less error the model had in predicting task completion. As shown in Figure 1, our choice of an rbf kernel is validated since the SVM gives the best accuracy, precision, recall, and F-1 score of all kernels.

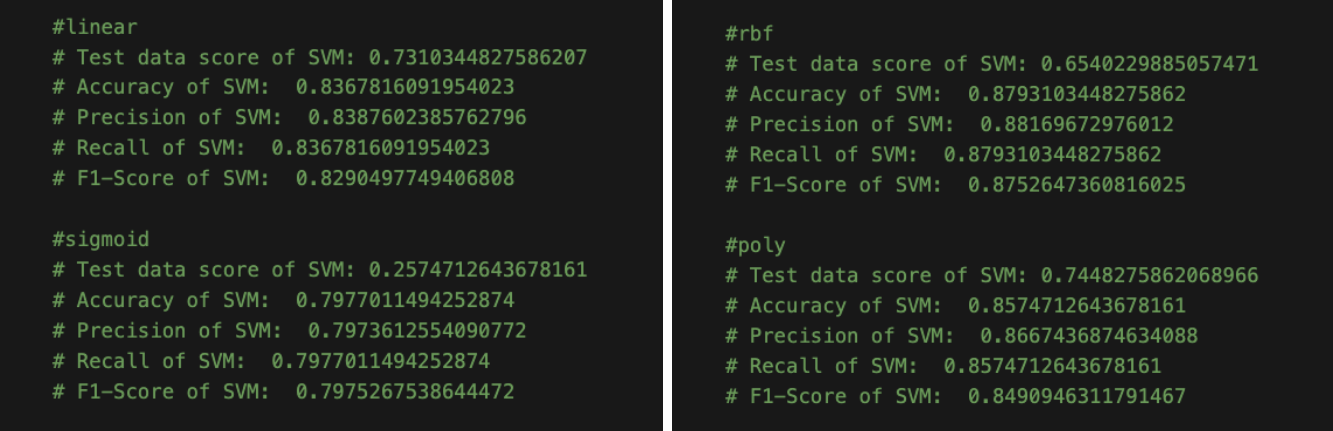
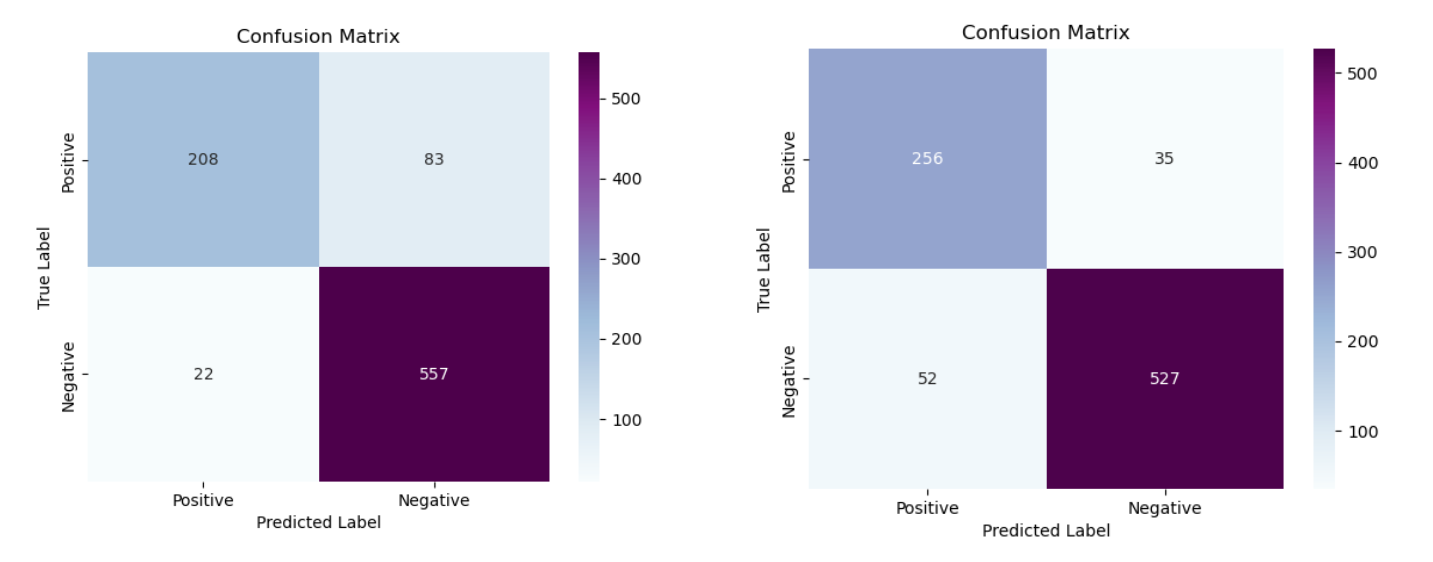
 ***Figure 1: SVM Kernels and Evaluation Metrics***

Figure 2 shows a comparison of prediction results between our SVM and K-Nearest Neighbors on a testing set in the form of confusion matrices. The results of these two models are comparable, where both models performed well at accurately classifying if a task was not completed. One noteworthy difference between the two models is that the SVM was able to identify more true negative results than the kNN model, where it could correctly classify 30 more uncompleted tasks than the kNN model. On the other hand, it mistakenly classified about 50 more tasks as incomplete than the kNN model. So, overall, our SVM could be seen as more “pessimistic” than the kNN model, since it will tend to classify more tasks as incomplete.

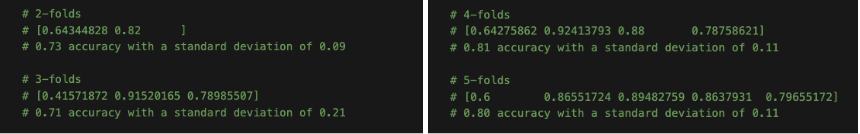
****

***Figure 2: Confusion Matrices: Support Vector Machine (left), K-Nearest Neighbors (right)***

Another way that we can evaluate our SVM classification model on unseen data is to use K-Folds cross-validation. This technique involves dividing the available data into multiple folds (subsets), using one fold for validation/test set and the remaining folds for training the model. The process is repeated K times where each time a different fold is used for the validation set. Results from each validation step are then averaged. This technique provides more than one test/train split of the input data, and provides a more robust estimate of the model’s performance. It divides the data into multiple folds and trains the SVM model on each fold while testing on the others.

The main purpose of cross-validation is to help prevent overfitting of the trained subset data to the model. New unseen data from our validation set could perform poorly on the model producing invalid predictions if the trained subset fit too well to the model. Good results from cross-validation will ensure that the model is robust and generalizes well to new data.

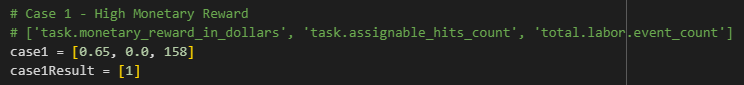
Testing with our SVM model we chose a 30% validation set and 70% training subset; having a higher percentage for the training subset will ensure that our input bias is minimized as this will include more relevant input features needed to create a more robust model. Figure 3 shows K-Fold cross validation metrics where the mean accuracy of the model gets better (closer to 1) at 4 or 5 folds. With the SVM model with kernel=’rbf’ and the one pass train/test split, we generated an accuracy of 0.879 which is better than the 4 or 5 fold results but they are similar to show that the model is robust. Setting K-Folds higher than 5, we didn’t notice the mean accuracy change much.



***Figure 3: K-Fold Cross Validation Metrics***

### **Model Analysis - 3 Cases of Interest**

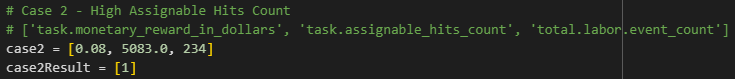
With cases of interest, we were interested to see how the SVM model would predict 3 new cases of interest based on the following variables: high monetary reward, high assignable HIT counts, and a high labor event count. These are the results we found when examining it.





***Figure 4: Interest Case 1 - Inputs & Model Result***

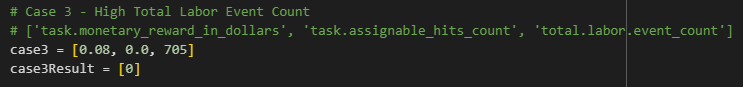
As shown in Figure 4, when examining this first case we can see a high amount of monetary rewards (in dollars) offered, along with a low amount of assignable hits and 158 total labor events that occurred within this case. As we can see this task was completed, which our model correctly predicted.





***Figure 5: Interest Case 2 - Inputs & Model Result***

As shown in Figure 5, when examining this second case we can see a high amount of assignable hits count offered, along with $0.08 of monetary reward and 234 total labor events that occurred within this case. As we can see this task was completed, which our model correctly predicted.





***Figure 6: Interest Case 3 - Inputs & Model Result***

As shown in Figure 6, when examining this third case we can see a high amount of total labor events occur, along with 0.0 amount of assignable hits and 705 total labor events. As we can see this task was not completed, which our model correctly predicted.