Task: Predicting how empathetic he or she is: Strongly disagree 1-2-3-4-5 Strongly agree

Dataset: Young people survey dataset from Kaggle
link: https://www.kaggle.com/miroslavsabo/young-people-survey
About the dataset: 1010 rows x 150 columns (139 integer and 11 categorical)
Target column: 'Empathy'
Preprocessing steps

- Drop the rows with NaN values in Empathy column (5 rows dropped)
- For all the Integer columns
  - Fill the NaN values with mean values for the column
- For categorical columns
  - Fill the NaN values with mode of the categorical columns. (statistics.mode function)
  - Use Sklearn.LabelEncoder() to encode the categorical columns into integer columns
- Split the dataset into Training (80%) and Testing (20%) sets (Train_Test_Split)

Baseline Model

- Perform the preprocessing steps as mentioned above
- Use Linear SVM classifier with 'OVR' method to fit the model and predict the labels (Acc = 30%)

Proposed Model

- Instead of selecting all the 149 columns as features the proposed model uses Pearson's Pearson product-moment correlation coefficients to pick the best features contributing to the target label.
  - Find the correlation coefficients of every column with respect to 'Empathy' column.
  - Sort the (column, value) pairs and pick the top n columns from the dictionary. The idea being these columns are highly correlated with the Target column (tried n=10,20)
- Drop all the other columns from the dataframe.
- Split the data into Train and Test datasets as mentioned above.
- Use SVM classifier with 'OVR' method to fit the model and predict the labels.
- Use Standard Scaling for preprocessing the feature values and GridsearchCV for parameter tuning.
- Evaluation: Accuracy and classification_report

The reason for choosing the proposed model was to let the model pick the top 10 features using a statistical test. Pearson Correlation coefficient provides the values for correlation of every feature with Target label. This helps the model learn from the best features contributing to the Target label without being handpicked manually.

Programming details:
Language: Python; Packages: Sklearn, Pandas, NumPy, Matplotlib, Seaborn, Statistics
Methods tried:
Tried various linear classifiers like Logistic Regression, Decision Trees, KNN and found that SVM performs the best. Hence used it for the baseline model.
StandardScaler to change the values of each column such that their mean is 0 and variance is 1. This helps the model learn the features better and is suggested in the sklearn documentation.
GridsearchCV for hyperparameter tuning.
Results

	Baseline model				Proposed Model (Top 10 features)				Proposed Model (Top 20 features)			
Labels	precision	recall	fscore	accuracy	precision	recall	fscore	accuracy	precision	recall	fscore	accuracy
1	0.2	0.67	0.31		0	0	0		0	0	0	
2	0.1	0.33	0.15		0	0	0		0.05	**0.33**	**0.09**	
3	0.01	0.33	0.03	41.05	0.31	0.41	0.35	**50.99**	0.19	0.37	0.25	49.66
4	0.57	0.29	0.38		0.51	0.36	0.42		0.51	0.37	0.43	
5	0.61	0.55	0.58		0.76	0.66	0.70		0.78	0.62	0.69	
avg/total	0.57	0.41	0.46		0.59	0.51	0.54		0.60	0.50	0.54	

Since target class 1 and 2 for 'Empathy' are very sparse, hence its very hard for the model to learn it correctly.
Hence, we observe that model considering top 10 features does not predict the sparse classes at all.
Increasing the number of features to 20 helps learn the sparse classes just a bit better. Providing some kind of attention to each of the classes in the Target Column and learning them separately, and picking the best prediction from all the 5 learnt models on the test set.