Machine Learning Assignment 2

Shrenik Doshi, MSc. AI

Abstract This project is done as the 2nd assignment for Machine learning module. It provides a basic flow of any machine learning task. From selection of dataset to finally select the best suited model for your data. The main objective of this project is to pre-process the dataset and research on any one pre-processing technique. Pre-processing of any dataset undergoes multiple steps which are discussed and implemented briefly further in the assignment. Top 3 models are selected after pre-processing of dataset and further hyper-parameter tuning is implemented so as to check the performance of each of the top models with a wide range of different parameter values. While performing different encoding techniques, it was observed that one-hot-encoding achieved highest accuracy but the performance and execution time of the model reduced and tend to reduce more if the unique data in the column increases. Binary Encoding is implemented with a motivation of learning new encoding technique.

Introduction

The dataset chosen for this project is an adult's income dataset. This dataset is downloaded from kaggle. Kaggle is a website where we can find multiple machine learning projects and competitions. There are a wide range of different dataset also available on this website. This dataset is one of those dataset from kaggle. The dataset consists of around 48000 rows and 16 columns. The target field is the **"income"** column. This column has 2 classes; **(1)** Income less than or equal to 50K (<=50K). **(2)** income greater than 50K (>50K). So this is basically a classification data. This data shows that an individual's annual income results from various factors which is influenced by the individual's education level, age, gender, occupation, and etc. For the purpose of considering all aspects of the assignment the dataset is reduced by filtering out some of the rows based on a column value which reduced the size of the dataset. This is discussed briefly ahead. Multiple machine learning models are implemented using scikit-learn (python library). The main motivation behind selecting this dataset is that it is a classification dataset. And also, to showcase the power of machine learning which can predict the individual's income just on the basis of the demographics of that individual. The main objective of the study is to learn and implement various machine learning models and evaluate their performance based on various hyper-parameters tuning. Further, as a research different feature encoding techniques will be implemented and evaluated as there are lots of categorical data columns. Some of the columns are very highly skewed, which, if handled and encoded properly can give better accuracy. While detecting the outlier, it was observed that most of the columns where categorical and encoded, so only 2 columns were available for outlier detection. The goal of this machine learning project is **to predict whether a person makes over 50K a year** or not given their demographic variation. To achieve this, several classification techniques are explored. Various machine learning models that are taught in the class or not taught have been implemented. The main focus of this project is to design and develop a model which is efficient, robust, reliable and as accurate as possible.

Research

There is a wide scope of research in the field of machine learning. In this dataset, the data consists of majority categorical data. The categorical columns can be converted to numerical form using different encoding techniques. Below are some of the encoding techniques which can be used in this scenario:

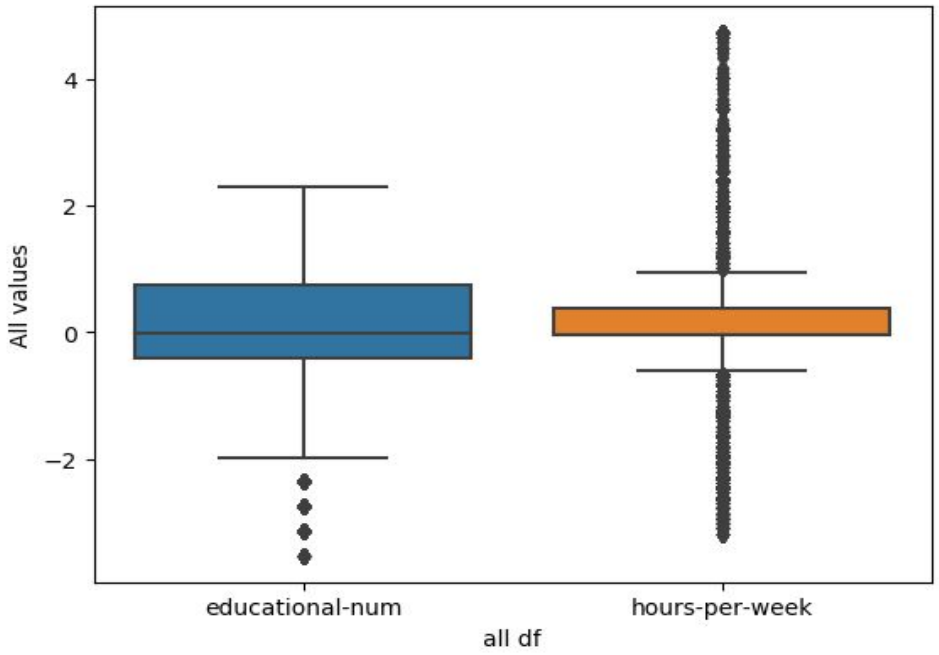
1. **Label Encoder**: Assign number labels to each unique values.
2. **Binary Encoder**: Binary encoding convert a category into a binary digits. Each binary digit creates one feature column.
3. **One-hot-encoding**: We map each category value to a vector that contains 1 and 0 denoting the presence or absence of the feature. The number of vectors depends on the number of categories for a features.
4. **Ordinal Encoding**: This encoding looks almost similar to Label Encoding but slightly different as Label coding would not consider whether variable is ordinal or not and it will assign sequence of integers
5. **Frequency Encoding**: It is a way to utilize the frequency of the categories as labels.

The columns like capital-gain, capital-loss and native-country has highly skewed data. A simple type of encoding is used where the data with highest occurence is assigned 1 and others are 0. For example, the column native-country has ~80% values are 'United States' and the rest are mixed countries. So when binary encoding is applied then I assign column value 1 when the country name is 'United States' and 0 for rest. Same process for capital-gain and capital-loss column, 1 assigned to all values which are non-zero. This process can gradually affect the accuracy of the models. To improve the accuracy, different encoding techniques which were taught in the class were implemented. Some of these techniques which will be part of research are: "LabelEncoder" and "One-hot-encoding". LabelEncoder technique was implemented which provided better results than before but are not that impressive. When "One-hot-encoding" was applied, the problem of **"Curse of dimensionality (CoD)"** arrised as this technique adds a new column for every unique values in the column but it provided better results than label encoding. The accuracy achieved by label encoding might be greater but it may cause more dimensionality which may slow down the model performance. Finally a technique known as **binary encoding** was implemented for columns like native-country, capital-loss and capital-gain. The problem of CoD is solved by this encoding technique. Implementing, learning and going deeper into this technique is the main area of research and it hasn't been taught in the class.

Methodology

The dataset chosen is related to adult income. In this dataset, there is a lot of junk values like '?' appeared in many of the columns and also some of the columns have very skewed data. The dataset contains more than 48000 rows. To accommodate everything properly and run all the hyper-parameter tuning test the dataset is initially reduced to around 13000 rows. There is a column named "**occupation**" and this column has many unique values. So to reduce the data only the rows having following values are removed filtered out: "Adm-clerical", "Craft-repair", "Exec-managerial", "Farming-fishing", "Other-service", "Priv-house-serv", "Protective-serv", "Sales", "Transport-moving". Filtering these data also reduces the chances of feature explosion. Secondly, there is a column called "age", there are lots of unique values, so I have created bins of different age groups. For example, if a person's age is 23 there he belows to age group of '10-30'. Following this step, dealing with missing values is done. For this step, firstly all the values which are '?' are converted to NaN and then these NaN values are replaced with the most frequent values in that particular column using SimpleImputer class. Once the missing are filled, is our data is ready for classification? No. For classification, all the data column values should be in numerical format. There are a lot of columns havings categorical data which are not suitable for classification. So we have to deal with this categorical data. For dealing with categorical data, many techniques can be implemented. In this assignment, One-hot-encoding is applied on some columns, i.e 'workclass', 'education', 'marital-status', 'occupation', 'relationship', 'race' and 'age-group'. Even label encoding is applied on some columns like gender, native-country, capital-gain and capital-loss. Now, we have all the data in the numerical but the data might have different value ranges. For eg. in column the range is from 10-100 and in other the range can be 1000-1000000. These differences in ranges might cause problem while classification. So we perform data scaling and data standardization. This process will make sure all the data is in the standard form. The next step is feature selection. Feature selection is not possible as most

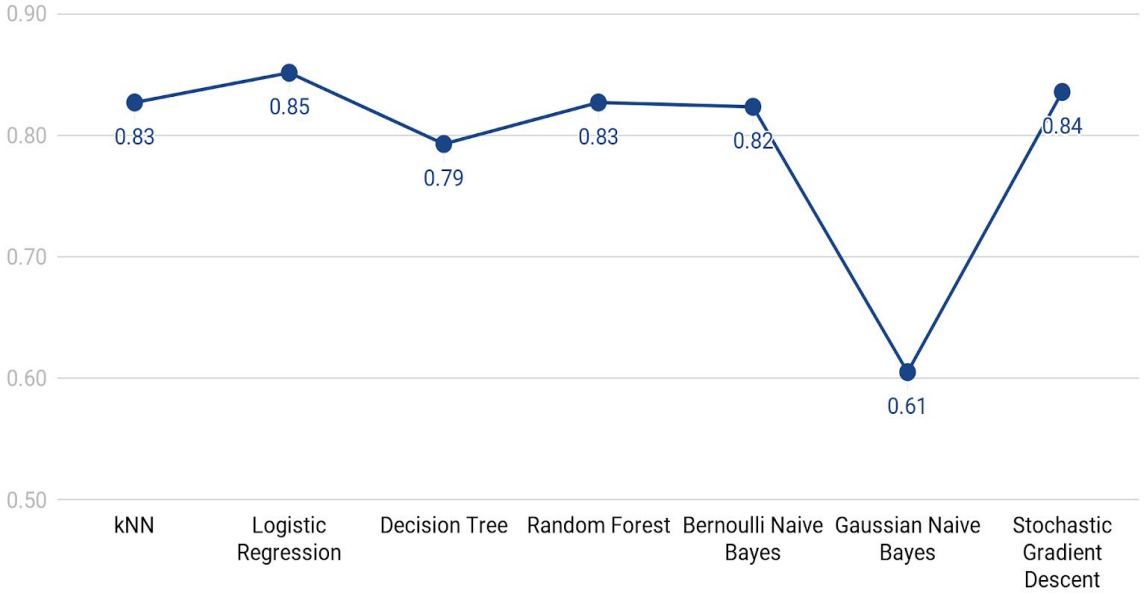
of the columns are categorical columns and are converted using one-hot-encoding or label encondizer. So performing feature selection is not appropriate in such cases. Once all the data is standardized, the next step is outlier detection. An outlier is an observation point that is distant from other observations. These exceptional data points can highly affect the performance and accuracy of the models. So to avoid this, outliers needs to be detected and handled. There are multiple techniques which can be used to detect outliers in the dataset. Below is the image of boxplot for some columns.



As there are outliers in these 2 columns but we cannot consider them outliers as these outliers are very close to each other, means they depict some information. As hours-per-week is an important column and contains very essential inform, so considering it as an outlier might give us a biased output. This output can be positive or negative. So we will not perform any outlier handling technique. This is the final stage in data pre-processing.

Evaluation

After pre-processing, we will build a wide range of models with their default parameters. While evaluating, we thoroughly test our models. Cross-fold validation is used to make sure that the model is not build on the less accurate data. Total of 5 folds are used in this process. The accuracy of model varies for each cross-fold step. For cross-fold validation, scikitlearn's GridSearchCV is used. The models built in this assignment are K-Nearest Neighbours, Logistic regression, decision tree classifier, random forest classifier, bernoulli naive bayes, gaussian naive bayes and stochastic gradient descent (SGD) Below is the accuracy of 8 models with their default parameters. Initially these models are evaluated on the basis of their default parameters. This will help us understand the basic accuracy that can be achieved with changing any parameters of the classifier. Kindly check below graph. This graph shows us different classification models implemented on the x-axis. On the y-axis, it is the accuracy for each model. Total seven machine learning models are implemented. All these models have different operation of calculating and predicting the target class. Bernoulli and Gaussian are 2 naive bayes classification models. It has been stated in the description of dataset on kaggle that this dataset is best suited and predicted by k-Nearest Neighbour classifier.



While using random forest classifier it was observed that increasing the number of trees does not affect the accuracy but if you increase the depth it affects the accuracy.

Number trees	Depth	Accuracy
100	2	0.77
100	25	0.83
200	25	0.83

The above table proves that increasing number of trees does not affect accuracy, but increasing depth improves accuracy. After standardization stage, the data may also contain negative values. So we cannot use multinomial naive bayes classifier as it does not support negative feature values. After building this model we select top 3 performing models and perform hyper-parameter tuning. We test these top models by tuning their different parameters and see the accuracy. All the models have different hyper-parameters and can be altered by many different possible values. Tuning these parameters may result in better accuracy or even accuracy can degrade. The top 3 models and their best parameters after hyper-parameter tuning are as below:

	Logistic Regression	SGD	KNN
Best parameters	C= 0.233 penalty=l2 solver=newton-cg	loss=log penalty=elasticnet	algorithm=brute, n_neighbors= 50, p:=1

From the above table shows the best parameters which resulted best accuracy for each model and the chart depicts accuracy of each model. From the above result we can say that, even tuning the model with different parameters the accuracy does not tend to change. The chart depicts that **LogisticRegression** gave the best accuracy after tuning of parameters. Below is the **confusion matrix** for the LogisticRegression model.

		Predicted	
		<=50K	**>50K**
Actual	**<=50K**	10626	926
	>50K	1397	2587

Below is the classification report having precision, recall and f1 score for the same model:

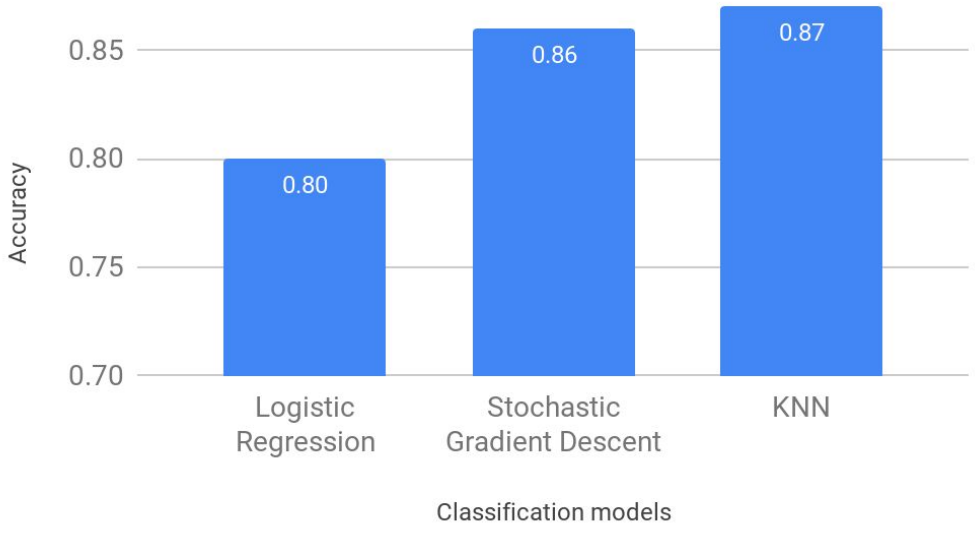
Average Precision Score	0.89
Average Recall Score	0.92
Average F1 score	0.9

The code calculates precision, recall and f1 score for each of the 3 top models, but for the purpose of showing experiment, these scores are displayed for just one model. The reduction in accuracy might be improper feature encoding as there are many categorical features in the data. So now, we go deeper into feature encoding process. We will try to evaluate accuracies when different encoding techniques are used. These encoding techniques can have a major impact on the performance of the model. The encoding techniques used during the research are: label encoding, one-hot-encoding and binary encoding on the columns such as native-country, capital-loss and capital-gain.

After label encoding:

We can see that after applying this technique, the best parameters got changed for all the 3 models. The accuracy of LogisticRegression decreased while accuracy increased for other two models.

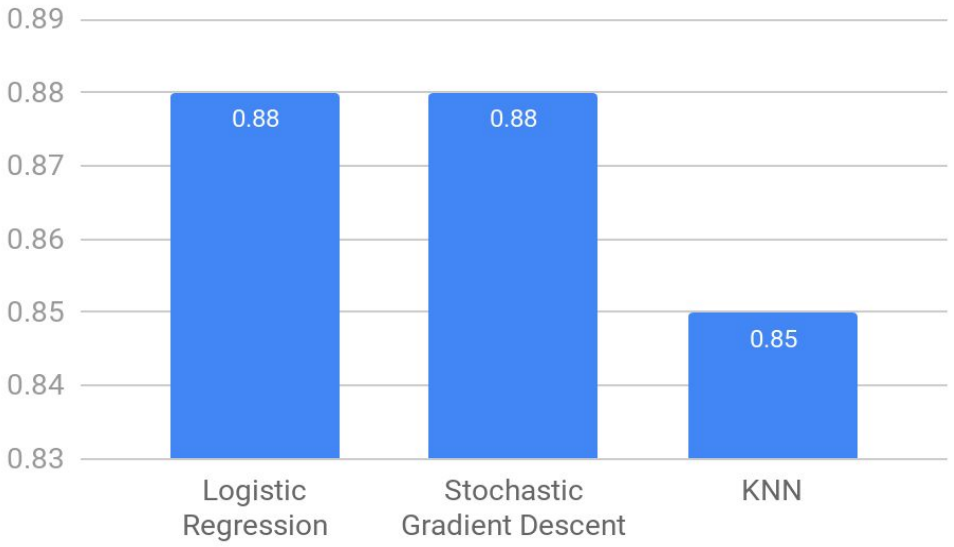
	Logistic Regression	Stochastic Gradient Descent	KNN
Best parameters	C= 0.6158 penalty=l2 solver=saga	loss=hinge penalty=elasticnet	algorithm=brute, n_neighbors= 30, p:=2



After one-hot-encoding:

Below is the accuracy when one-hot-encoder is used. Accuracy is much higher than the previous methods. While one-hot encoding solves the problem of unequal weights given to categories within a feature, it is not very useful when there are many categories, as that will result in the formation of as many new columns, which can result in the curse of dimensionality. The concept of the “curse of dimensionality” discusses that in high-dimensional spaces some things just stop working properly.

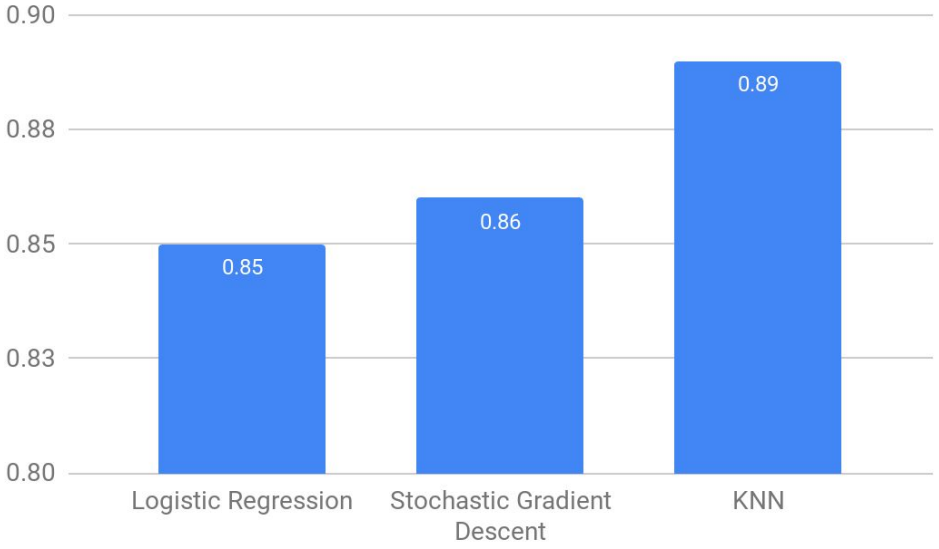
	Logistic Regression	SGD	KNN
Best parameters	C= 206.9138 penalty=l2 solver=newton-cg	loss=log penalty=elasticnet	algorithm=auto, n_neighbors= 45, p:=1



After binary encoding:

The below are the results after applying binary encoding on the top 3 models. It is observed that, kNN model gave the best accuracy as compared to other 2. This accuracy is the highest accuracy across all the project.

	Logistic Regression	SGD	KNN
Best parameters	C= 0.08858 penalty=l2 solver=saga	loss=log penalty=elasticnet	algorithm=auto, n_neighbors= 45, p:=1



This is how accuracy varies when we use different encoding methods for these categorical columns. We can notice that maximum accuracy achieved is from kNN when used with binary encoding method.

Conclusion

At the end, I have learned different machine learning models and got to know more about scikit-learn which is a python library. Implementing hyper-parameter tuning after selecting the best performing models resulted in improvement of accuracies for that models. As a research topic, different feature encoding techniques were used. Using some techniques taught in the class and some techniques not taught got me learn more about feature selection. There are a vast variety of techniques that can be used for feature encoding. In future, I will implement mean encoding, frequency encoding and hash encoding.