Inconometrics

A Salary prediction model

Algorithm: Naive Bayes

Presented by:

Ajay Deshmukh

Darshan Bhansali

Vishwadeep Dutta

Introduction:

Inconometrics is a digital market company. It promotes its products based on the customer data it generates every year. So this year Inconometrics is in the requirement to predict the salary of the customer using various metrics available to them. They will use this salary prediction feature to recommend their various products. They have defined their product into categories Economic Product and Premium Product.

The category is based on its price and yearly maintenance fees. A person with a salary below $50K would find it uncomfortable to afford the Premium products and the Premium product comes with various Add-on services and with those add-on services the user can leverage the full-feature of the product. Whereas, the economic product has all the basic features under an economic price range which will not affect the budget of our economic customers.

Our main agenda here is to only present the best-suited product for the customer, rather than confusing them or spamming them with products that may not meet their expectations.

Marketing Analysts of Inconometrics have predicted that depending on the accuracy of our Salary prediction model and suggestions to the customer. We can increase our revenue by 10% for the coming quarter.

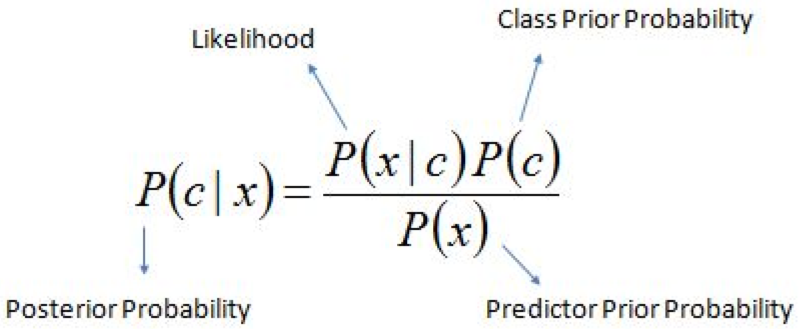
The motivation behind this was not to spam the customer with all the products available from us. “We work on the principle of going for quality rather than quantity”.

In Inconometrics, we conducted a survey for our customer where they were asked to provide the suggestion of products they like and they survey showed that most people with a salary below 50K are going for economic products and people who are having salary above 50K are providing a strong liking for our Premium products and also expressed their interest in the Add-ons.

Algorithm- Naive Bayes:

In machine learning, naïve Bayes classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naïve) independence assumptions between the features. This Algorithm is called “Naive” because it makes a naive assumption that each feature is independent of other features which is not true in real life.

Bayes theorem provides a way of calculating the posterior probability, $P(c|x)$, from $P(c)$, $P(x)$, and $P(x|c)$. Naive Bayes classifiers assume that the effect of the value of a predictor (x) on a given class (*c*) is independent of the values of other predictors. This assumption is called class conditional independence.



$$P(c\,|\,\mathrm{X}) = P(x_1\,|\,c)\times P(x_2\,|\,c)\times\cdots\times P(x_n\,|\,c)\times P(c)$$

- *P*(*c*|*x*) is the posterior probability of *class* (*target*) given *predictor* (*attribute*).
- *P*(*c*) is the prior probability of *class*.
- *P*(*x*|*c*) is the likelihood which is the probability of a predictor given *class*.
- *P*(*x*) is the prior probability of the predictor.

Data Cleaning:

- We have taken an adult dataset from the Kaggle (https://archive.ics.uci.edu/ml/datasets/census+income) which provides data about the customer's attributes and their salary to train our models. Our data shape has 15 columns and 48562 rows.
- We have dropped 3554 rows from our dataset as these rows had garbage values especially with "?".

- Columns data types were of object type so converted each column data to Integer or String based on underlying data. Following is a table shows our columns after conversion.

age	Integer	Continuous

Workclass	String	Categorical
fnlwgt	Integer	Continuous
Education	String	Categorical
education_num	Integer	Continuous
marital_status	String	Categorical
occupation	String	Categorical
relationship	String	Categorical
race	String	Categorical
gender	String	Categorical
capital_gain	Integer	Continuous
capital_loss	Integer	Continuous
hours_per_week	Integer	Continuous
native_country	String	Categorical
income_bracket	String	Categorical

- We have dropped the capital gain and capital loss columns because they have mostly zero values which are of no use. The education-num column is derived from the

Education column. In naive bayes we consider all the features to be independent variables thus to normalize the weightage of the column, we have dropped it.

- For preprocessing of data we have converted continuous values to categorical values as Naive bayes works only with categorical values. We have converted age and hours per week column to categorical column by creating bins for respective data.

- Naive Bayes works best with less categorical values so have re-classified all the columns categories into lesser number of more accurate categories.

Models:

We have implemented following 4 models to find which model gives the best accuracy score:

1. Gaussian Naive Bayes: Sklearn
2. Multinomial Naive Bayes: Sklearn
3. Categorical Naive Bayes: Sklearn
4. Multinomial Naive Bayes : Spark ML

Multinomial Naive Bayes : Spark ML

Model introduction:

We use multinomial classifiers where we have to predict more than 2 classes. Here we are predicting whether the salary is greater than 50k or less than 50k. So we are predicting two classes that is why we have used a multinomial Naive Bayes algorithm.

Algorithm Summary

- Input: classLabels (>50k, <50k), feature vectors (values from all the columns which are used in fitting of model are stored in a single array)
- Assumptions: Independence between every pair of features, Feature values are nonnegative, such as counts

Approach:

- Created a SQL table from pandas dataframe to fit in the model
- Selected columns all the columns except native countries to fit in the algorithm
- Splitted the data into test and train data set by 70:30 ratio
- Converted the strings categories into numerical categories using string indexer because multinomial naive bayes works on numerical categorical values

- Used the VectorAssembler() to merge our feature columns into a single vector column.
- Created a Multiclass Naive Bayes Classifier.

Model Fitting:

- Import *from pyspark.ml.classification import NaiveBayes, from pyspark.ml import Pipeline*
- For fitting the model here for this model we distributed the data in the test and train model in the 70:30 ratio
- Created a Multinomial Naive Bayes Classification Model with the variable var_smoothing=0. the model and predicted results.
- After model fitting we created one dataframe named as prediction which contains all the prediction values of test data

Model Evaluation:

Used Confusion Matrix to evaluate the model by finding the True positives and True negatives. The Confusion Matrix was used to calculate the accuracy of the model which came out to be **73.9%**. Used multiple matrices to analyze the results where the precision, recall, F1 score came out to be 73.9%.

Gaussian Naïve Bayes:Scikit-learn

Model Introduction

In Gaussian Naive Bayes, continuous values associated with each feature are assumed to be distributed according to a Gaussian distribution. Gaussian Naive Bayes is best suited for continuous data, but it can be applied to categorical data as well. A Gaussian distribution is also called Normal distribution.

Approach:

- The data manipulation for Naïve Bayes was a bit different and the steps included importing raw data directly from the CSV file to a dataframe namely ‘dfg’.
- We converted the categorical columns of the data set into continuous integer data.
- Verified through charts that some of the features follow Gaussian distribution. (Marital Status, education, work class, and occupation). Thus, we concluded that the Gaussian Naïve Bayes model can be implemented on this dataset.

- Dropped the column of ‘fnlwgt’ because the column neither has categorical data nor does the continuous data follow a gaussian distribution.

Model fitting and Tools used:

- Import module ‘train_test_split ‘ from library ‘sklearn.model_selection’ to split the training dataset into train and test data.
- Split the given dataset into training and test dataset in the ratio of 80:20. Defined the features to be used for training the model.
- Created a Gaussian Naive Bayes Classification Model with the variable var_smoothing=0.
- Trained the model and predicted results.

Model Evaluation:

Used Confusion Matrix to evaluate the model by finding the True positives and True negatives. The Confusion Matrix was used to calculate the accuracy of the model which came out to be **73.83%**. Developed the classification report to analyze the results where the precision came out to be 0.88 and recall being 0.75.

Multinomial Naïve Bayes:Scikit-learn

Model Introduction:

The multinomial Naive Bayes classifier is suitable for classification with discrete features (e.g., word counts for text classification). The multinomial distribution normally requires integer feature counts. However, in practice, fractional counts may also work.

Approach:

- Initially categorized the features like age, education, occupation, etc. to reduce the dimensionality.
- As the definition states for this model it works on classification with discrete values/features, thus converting the categorical data into discrete binary values for each feature.
- Used One-hot encoding to manipulate the data to create binary values for each feature category.

Model fitting and Tools used:

- Import module ‘train_test_split ‘ from library ‘sklearn.model_selection’ to split the training dataset into train and test data.

- Split the given dataset into training and test dataset in the ratio of 80:20. Defined the features to be used for training the model.
- Imported the 'MultinomialNB' from 'sklearn' and created a Multinomial Naive Bayes Classification Model.
- Trained the model and predicted results.

Model Evaluation:

Used Confusion Matrix to evaluate the model by finding the True positives and True negatives. The Confusion Matrix was used to calculate the accuracy of the model which came out to be **79.73%.** Developed the classification report to analyze the results where the precision came out to be 0.90 and recall being 0.82.

Categorical Naïve Bayes: Scikit-learn

Model Introduction:

CategoricalNB by sci-kit-learn is a new class to be added in the naive_bayes module. CategoricalNB implements the categorical naive Bayes algorithm for categorically distributed data. It considers the most basic assumption of the Naive Bayes i.e. features have discrete and categorical values. CategoricalNB assumes that the sample matrix X is encoded (for instance with the help of OrdinalEncoder) such that all categories for each feature are represented with numbers 0 to ni-1 where ni is the number of available categories of feature i.

Approach:

- Since the algorithm needs categorized values we categorized columns like age, workclass, education, marital status, occupation, Native country and hours per week to better categorize the parameters and reduce dimensionality.
- These columns were initially converted into continuous integer values and then used the method pd.Categorical() to convert these values into categorical values. Thus, converting categorical data into categorical integer data.
- Dropped the column of 'fnlwgt' because the column does not have categorical data.

Model Fitting and Tools Used:

- Import module 'train_test_split ' from library 'sklearn.model_selection' to split the training dataset into train and test data.

- Split the given dataset into training and test dataset in the ratio of 80:20. Defined the features to be used for training the model, set the variable rand_state=42 so that the results remain consistent.
- Imported the 'CategoricalNB' from 'sklearn' and created a Categorical Naive Bayes Classification Model.
- Trained the model and predicted results.

Model Evaluation:

Used Confusion Matrix to evaluate the model by finding the True positives and True negatives. The Confusion Matrix was used to calculate the accuracy of the model which came out to be **81.39%**. Developed the classification report to analyze the results where the precision came out to be 0.90 and recall being 0.85

Models Evaluation for real time use:

Since we have applied 4 different methodologies to predict the model for our organization. And ordering them from least to most accurate:

1. Gaussian Model Scikit-learn: 73.83%
2. Multinomial Model Spark ML: 73.9%
3. Multinomial Model Scikit-learn:79.74%
4. **Categorical Model Scikit-learn: 81.38%**

So as from the above ordering, the team has decided to implement the Categorical Model Scikit-learn model as it is giving the highest accuracy.

Important Insight from the ML model:

So from the following model, we can deduce that males belonging to the senior age group(45-65) working in a private class, having a Master's level education and carrying a marital status of being never married are having a maximum probability of earning more than 50k. Considering them as more **Workaholic**.

Notes :

1. For the categorical Naive Bayes, you will need the latest version of Sklearn library installed i.e. version 0.22.2
2. Command to check sklearn version: python -m pip show sci-kit-learn

References

1. *https://www.kaggle.com/johnolafenwa/us-census-data*
2. *https://archive.ics.uci.edu/ml/datasets/census+income*
3. *https://towardsdatascience.com/naive-bayes-intuition-and-implementation-ac328f9c9718*
4. *https://www.machinelearningplus.com/predictive-modeling/how-naive-bayes-algorithm-works-with-example-and-full-code/*
5. *https://www.ritchieng.com/machine-learning-evaluate-classification-model/*
6. *https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c*
7. *https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/3741049972324885/3783546674231736/4413065072037724/latest.html*
8. *https://sites.google.com/site/complexdataminingproject/*
9. *https://www.saedsayad.com/naive_bayesian.htm*
10. *http://cseweb.ucsd.edu/classes/sp15/cse190-c/reports/sp15/048.pdf*
11. *https://www.geeksforgeeks.org/naive-bayes-classifiers/*

12. *https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html*
13. *https://scikit-learn.org/stable/modules/naive_bayes.html*