# Naive Bayes

**Instructions:**

Please share your answers filled in-line in the word document. Submit code separately wherever applicable.

Please ensure you update all the details:

**Name: Vishvash C Batch ID:** 23012024

**Topic: Naïve Bayes**

**Guidelines:**

**1. An assignment submission is considered complete only when correct and executable code(s) are submitted along with the documentation explaining the method and results. Failing to submit either of those will be considered an invalid submission and will not be considered as correct submission.**

**2. Ensure that you submit your assignments correctly and in full. Resubmission is not allowed.**

**3. Post the submission you can evaluate your work by referring to keys provided. (will be available only post the submission).**

**Hints:**

1. **Business Problem**
   1. **What is the business objective?**
   2. **Are there any constraints?**
2. **Work on each feature of the dataset to create a data dictionary as displayed in the below image:**



**2.1 Make a table as shown above and provide information about the features such as its data type and its relevance to the model building. And if not relevant, provide reasons and a description of the feature.**

1. **Data Pre-processing**

**3.1 Data Cleaning, Feature Engineering, etc.**

1. **Exploratory Data Analysis (EDA):**
   1. **Summary.**
   2. **Univariate analysis.**
   3. **Bivariate analysis.**
2. **Model Building**
   1. **Build the model on the scaled data (try multiple options).**
   2. **Build a Naïve Bayes model.**

**5.3 Validate the model with test data and obtain a confusion matrix, get precision, recall, and accuracy from it.**

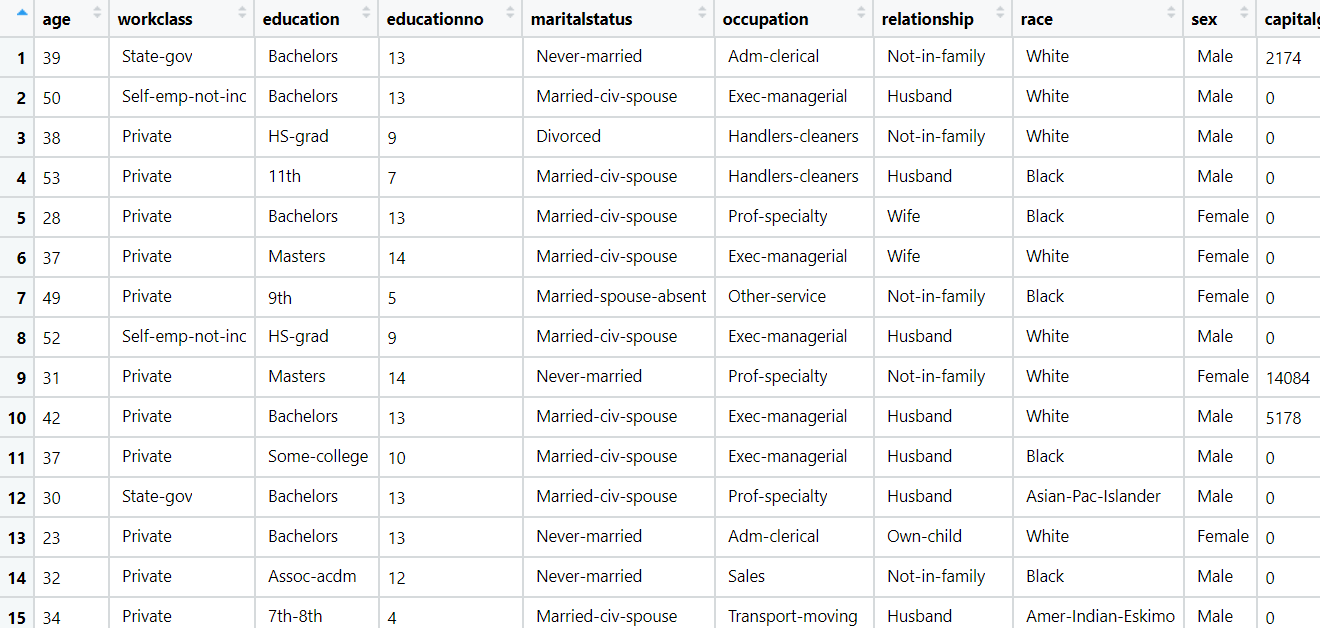
**5.4 Tune the model and improve the accuracy**

**6. Write about the benefits/impact of the solution - in what way does the business (client) benefit from the solution provided?**

**Problem Statement:**

Prepare a classification model using the Naive Bayes algorithm for the salary dataset. Train and test datasets are given separately. Use both for model building. And predict Salary

Note : Do the Deployment by using Flask Framework



|  |  |  |  |
| --- | --- | --- | --- |
| Name of Feature | Description | Type | Relevance |
| age | Age of the individual | Quantitative | Irrelevant |
| workclass | Type of work class | Nominal | Relevant |
| education | Level of education | Nominal | Relevant |
| educationno | Number of years of education | Quantitative | Irrelevant |
| maritalstatus | Marital status | Nominal | Relevant |
| occupation | Occupation | Nominal | Relevant |
| relationship | Relationship status | Nominal | Relevant |
| race | Race of the individual | Nominal | Relevant |
| sex | Gender | Nominal | Relevant |
| capitalgain | Capital gain | Quantitative | Irrelevant |
| capitalloss | Capital loss | Quantitative | Irrelevant |
| hoursperweek | Number of hours worked per week | Quantitative | Irrelevant |
| native | Native country | Nominal | Relevant |
| Salary | Salary level | Nominal | Relevant |

**Code:**

'''CRISP-ML(Q)

a. Business & Data Understanding

As the economy becomes more competitive, companies are increasingly relying on data-driven approaches to optimize their operations and make strategic decisions. One such area is predicting salary levels based on various demographic and job-related factors. Understanding the determinants of salary levels can help companies attract and retain talent, optimize compensation structures, and ensure fair pay practices.

i. Business Objective - Optimize Salary Prediction

ii. Business Constraint - Minimize Prediction Errors

Success Criteria:

1. Business Success Criteria - Increase employee satisfaction by 15% by accurately predicting salary levels and ensuring fair pay practices.

2. ML Success Criteria - Achieve a prediction accuracy of atleast by 80% and performance of predicting salary levels for new data.

3. Economic Success Criteria - Reduce turnover costs by accurately predicting salary levels and offering competitive compensation packages. By optimizing salary prediction, companies can reduce turnover rates by atleast 20 %.

Data Collection - Salary data from various companies and industries is collected, including information on age, education, occupation, work experience, and other relevant factors. The dataset contains 15 features and the target variable is salary level.

Metadata Description:

Feature Name Description Type Relevance

-------------- ----------------------------------------- ------------- --------------

age Age of the individual Quantitative Relevant

workclass Type of work class Nominal Relevant

education Level of education Nominal Relevant

educationno Number of years of education Quantitative Relevant

maritalstatus Marital status Nominal Relevant

occupation Occupation Nominal Relevant

relationship Relationship status Nominal Relevant

race Race of the individual Nominal Relevant

sex Gender Nominal Relevant

capitalgain Capital gain Quantitative Relevant

capitalloss Capital loss Quantitative Relevant

hoursperweek Number of hours worked per week Quantitative Relevant

native Native country Nominal Relevant

Salary Salary level Nominal Relevant

'''

# Code modularity must be maintained

# Import all the required libraries and modules

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

# imbalanced-learn pipeline is being called in rather than a scikit-learn one.

# This is because we will be using SMOTE in our pipeline.

# pip install imblearn

from imblearn.pipeline import make\_pipeline

from imblearn.over\_sampling import SMOTE

# SMOTE - Synthetic Minority Over-sampling Technique

from sklearn.naive\_bayes import MultinomialNB

from sklearn.feature\_extraction.text import CountVectorizer

import sklearn.metrics as skmet

import joblib

from sqlalchemy import create\_engine, text

from urllib.parse import quote

from sklearn.model\_selection import GridSearchCV

# Loading the data set

data1 = pd.read\_csv(r"C:/Users/Lenovo/Downloads/Study material/Data Science/naivebayes/Assignment/Naive Bayes/SalaryData\_Train.csv", encoding = "ISO-8859-1")

# Mapping the type to numeric values 1 and 0.

# This step is required for metric calculations in model evaluation phase.

data1.info()

data = data1.drop(columns = ["age", "educationno", "capitalgain", "capitalloss", "hoursperweek"])

data.info()

data['High\_Sal'] = np.where(data.Salary == ' >50K', 1, 0)

data.drop(columns = ["Salary"], inplace = True)

###############################################################################

# MYSQL

# pip install pymysql

conn\_string = ("mysql+pymysql://{user}:{pw}@localhost/{db}"

.format(user = "root", # user

pw = "1234", # password

db = "salary\_db")) # database

db = create\_engine(conn\_string)

###############################################################################

# PostgreSQL

# pip install psycopg2

# Creating engine which connect to postgreSQL

# conn\_string = psycopg2.connect(database = "postgres", user = 'postgres', password = 'monish1234', host = 'localhost', port= '5432')

# conn\_string = ("postgresql+psycopg2://{user}:{pw}@localhost/{db}"

# .format(user = "postgres", # user

# pw = quote("postgres"), # password

# db = "postgres")) # database

###############################################################################

db = create\_engine(conn\_string)

conn = db.connect()

data.to\_sql('salary\_raw', con = conn, if\_exists = 'replace', index = False)

conn.autocommit = True

###############################################################################

# Select query

sql = 'SELECT \* from salary\_raw'

salary\_data = pd.read\_sql\_query(text(sql), conn)

# Data Preprocessing - textual data

# Imbalance check

salary\_data.High\_Sal.value\_counts()

salary\_data.High\_Sal.value\_counts() / len(salary\_data.High\_Sal) # values in percentages

# alternatively

salary\_data.groupby(['High\_Sal']).size()

salary\_data.groupby(['High\_Sal']).size() / len(salary\_data.High\_Sal)

# Data Split

salary\_train, salary\_test = train\_test\_split(salary\_data, test\_size = 0.2, stratify = salary\_data[['High\_Sal']], random\_state = 0) # StratifiedKFold is a variation of k-fold which returns stratified folds: each set contains approximately the same percentage of samples of each target class as the complete set.

# CountVectorizer

# Convert a collection of text documents to a matrix of token counts

countvectorizer = CountVectorizer(analyzer = 'word', stop\_words = 'english')

###########################

# for illustrative purposes

# s\_sample = salary\_train.loc[salary\_train.text.str.len() < 50].sample(3, random\_state = 35)

# s\_sample = s\_sample.iloc[:, 0:2]

# # Document Term Matrix with CountVectorizer (# returns 1D array)

# s\_vec = pd.DataFrame(countvectorizer.fit\_transform(s\_sample.values.ravel()).\

# toarray(), columns = countvectorizer.get\_feature\_names\_out())

# s\_vec

###########################

# creating a matrix of token counts for the entire text document

def split\_into\_words(i):

return [word for word in i.split(" ")]

# Get a list of columns except the excluded column

columns\_to\_include = [col for col in salary\_data.columns if col != 'High\_Sal']

# Concatenate text from all columns except the excluded column into a single series

salary\_data\_combined\_text = ''

for col in columns\_to\_include:

salary\_data\_combined\_text += salary\_data[col] + ' '

salary\_train\_combined\_text = ''

for col in columns\_to\_include:

salary\_train\_combined\_text += salary\_train[col] + ' '

salary\_test\_combined\_text = ''

for col in columns\_to\_include:

salary\_test\_combined\_text += salary\_test[col] + ' '

# Defining the preparation of email texts into word count matrix format - Bag of Words

salary\_bow = CountVectorizer(analyzer = split\_into\_words).fit(salary\_data\_combined\_text)

# Defining BOW for all messages

all\_salary\_matrix = salary\_bow.transform(salary\_data\_combined\_text)

# For training messages

train\_salary\_matrix = salary\_bow.transform(salary\_train\_combined\_text)

# For testing messages

test\_salary\_matrix = salary\_bow.transform(salary\_test\_combined\_text)

# We will use SMOTE technique to handle class imbalance.

# Oversampling can be a good option when we have class imbalance.

# Due to this our model will perform poorly in capturing variation in a class

# because we have too few instances of that class, relative to one or more other classes.

# SMOTE: Is an approach is to oversample (duplicating examples) the minority class

# This is a type of data augmentation for the minority class and is referred

# to as the Synthetic Minority Oversampling Technique, or SMOTE for short.

smote = SMOTE(random\_state = 0)

# Transform the dataset

X\_train, y\_train = smote.fit\_resample(train\_salary\_matrix, salary\_train.High\_Sal)

y\_train.unique()

y\_train.values.sum() # Number of '1's

y\_train.size - y\_train.values.sum() # Number of '0's

# The data is now balanced

# Multinomial Naive Bayes

classifier\_mb = MultinomialNB() #vanilla model with default parameters

classifier\_mb.fit(X\_train, y\_train)

# Evaluation on Test Data

test\_pred\_m = classifier\_mb.predict(test\_salary\_matrix)

pd.crosstab(salary\_test.High\_Sal, test\_pred\_m)

# Accuracy

accuracy\_test\_m = np.mean(test\_pred\_m == salary\_test.High\_Sal)

accuracy\_test\_m

# or alternatively

skmet.accuracy\_score(salary\_test.High\_Sal, test\_pred\_m)

# Training Data accuracy

train\_pred\_m = classifier\_mb.predict(train\_salary\_matrix)

pd.crosstab(salary\_train.High\_Sal, train\_pred\_m)

# Accuracy

accuracy\_train\_m = np.mean(train\_pred\_m == salary\_train.High\_Sal)

accuracy\_train\_m

skmet.accuracy\_score(salary\_train.High\_Sal, train\_pred\_m)

############################################

# Model Tuning - Hyperparameter optimization

# Multinomial Naive Bayes changing default alpha for laplace smoothing

# if alpha = 0 then no smoothing is applied and the default alpha parameter is 1

# the smoothing process mainly solves the emergence of zero probability problem in the dataset.

# formula:

# P(w|High\_Sal) = (num of High\_Sal with w + alpha)/(Total num of High\_Sal salary + K(alpha))

# K = total num of words in the email to be classified

param\_grid = {

'alpha': [0.01, 0.1, 0.5, 1.0,5.0 , 10.0], # Additive smoothing parameter

'fit\_prior': [True, False], # Whether to learn class prior probabilities or not

}

clf = MultinomialNB()

# Initialize the GridSearchCV object

grid\_search = GridSearchCV(estimator=clf, param\_grid=param\_grid, cv=5, scoring='accuracy', verbose=1)

NB\_NEW = grid\_search.fit(X\_train, y\_train)

print(NB\_NEW.best\_params\_)

# Evaluation on Test Data after applying laplace

test\_pred\_lap = grid\_search.predict(test\_salary\_matrix)

pd.crosstab(test\_pred\_lap, salary\_test.High\_Sal)

accuracy\_test\_lap = np.mean(test\_pred\_lap == salary\_test.High\_Sal)

accuracy\_test\_lap

skmet.accuracy\_score(salary\_test.High\_Sal, test\_pred\_lap)

# Training Data accuracy

train\_pred\_lap = grid\_search.predict(train\_salary\_matrix)

pd.crosstab(train\_pred\_lap, salary\_train.High\_Sal)

accuracy\_train\_lap = np.mean(train\_pred\_lap == salary\_train.High\_Sal)

accuracy\_train\_lap

skmet.accuracy\_score(salary\_train.High\_Sal, train\_pred\_lap)

# Metrics

print("accuracy: %.2f, sensitivity: %.2f, specificity: %.2f, precision: %.2f" %

(skmet.accuracy\_score(salary\_test.High\_Sal.ravel(), test\_pred\_lap),

skmet.recall\_score(salary\_test.High\_Sal.ravel(), test\_pred\_lap),

skmet.recall\_score(salary\_test.High\_Sal.ravel(), test\_pred\_lap, pos\_label = 0),

skmet.precision\_score(salary\_test.High\_Sal.ravel(), test\_pred\_lap)))

# Confusion Matrix - Heat Map

cm = skmet.confusion\_matrix(salary\_test.High\_Sal, test\_pred\_lap)

cmplot = skmet.ConfusionMatrixDisplay(confusion\_matrix = cm, display\_labels = ['Not High\_Sal', 'High\_Sal'])

cmplot.plot()

cmplot.ax\_.set(title = 'High\_Sal Detection Confusion Matrix',

xlabel = 'Predicted Value', ylabel = 'Actual Value')

# Saving the Best Model using Pipelines

# Building the Pipeline

# Defining Pipeline

pipe1 = make\_pipeline(countvectorizer, smote, grid\_search)

# Fit the train data

processed = pipe1.fit(salary\_train\_combined\_text.ravel(), salary\_train.High\_Sal.ravel())

# Save the trained model

joblib.dump(processed, 'processed1')

# load the saved model for predictions

model = joblib.load('processed1')

# Predictions

test\_pred = model.predict(salary\_test\_combined\_text.ravel())

# Evaluation on Test Data with Metrics

# Confusion Matrix

pd.crosstab(salary\_test.High\_Sal, test\_pred)

# Accuracy

skmet.accuracy\_score(salary\_test.High\_Sal, test\_pred)

# Metrics

print("accuracy: %.2f, sensitivity: %.2f, specificity: %.2f, precision: %.2f" %

(skmet.accuracy\_score(salary\_test.High\_Sal.ravel(), test\_pred),

skmet.recall\_score(salary\_test.High\_Sal.ravel(), test\_pred),

skmet.recall\_score(salary\_test.High\_Sal.ravel(), test\_pred, pos\_label = 0),

skmet.precision\_score(salary\_test.High\_Sal.ravel(), test\_pred)))

# Confusion Matrix - Heat Map

cm = skmet.confusion\_matrix(salary\_test.High\_Sal, test\_pred)

cmplot = skmet.ConfusionMatrixDisplay(confusion\_matrix = cm, display\_labels = ['Not High\_Sal', 'High\_Sal'])

cmplot.plot()

cmplot.ax\_.set(title = 'High\_Sal Detection Confusion Matrix',

xlabel = 'Predicted Value', ylabel = 'Actual Value')

**Output:**

columns\_to\_include

Out[29]:

['workclass',

'education',

'maritalstatus',

'occupation',

'relationship',

'race',

'sex',

'native']

combined\_text

Out[32]:

0 State-gov Bachelors Never-married Adm-cler...

1 Self-emp-not-inc Bachelors Married-civ-spou...

2 Private HS-grad Divorced Handlers-cleaners...

3 Private 11th Married-civ-spouse Handlers-c...

4 Private Bachelors Married-civ-spouse Prof-...

2995 Private Bachelors Married-civ-spouse Exec-...

2996 Private Bachelors Married-civ-spouse Tech-...

2997 Private HS-grad Married-civ-spouse Prof-sp...

2998 Private Bachelors Never-married Prof-speci...

2999 Private 10th Widowed Other-service Not-in...

Name: workclass, Length: 3000, dtype: object

pd.crosstab(salary\_test.High\_Sal, test\_pred\_m)

Out[56]:

col\_0 0 1

High\_Sal

0 346 99

1. 35 120

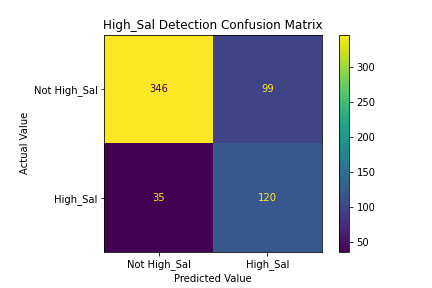
accuracy\_test\_m

Out[59]: 0.7766666666666666

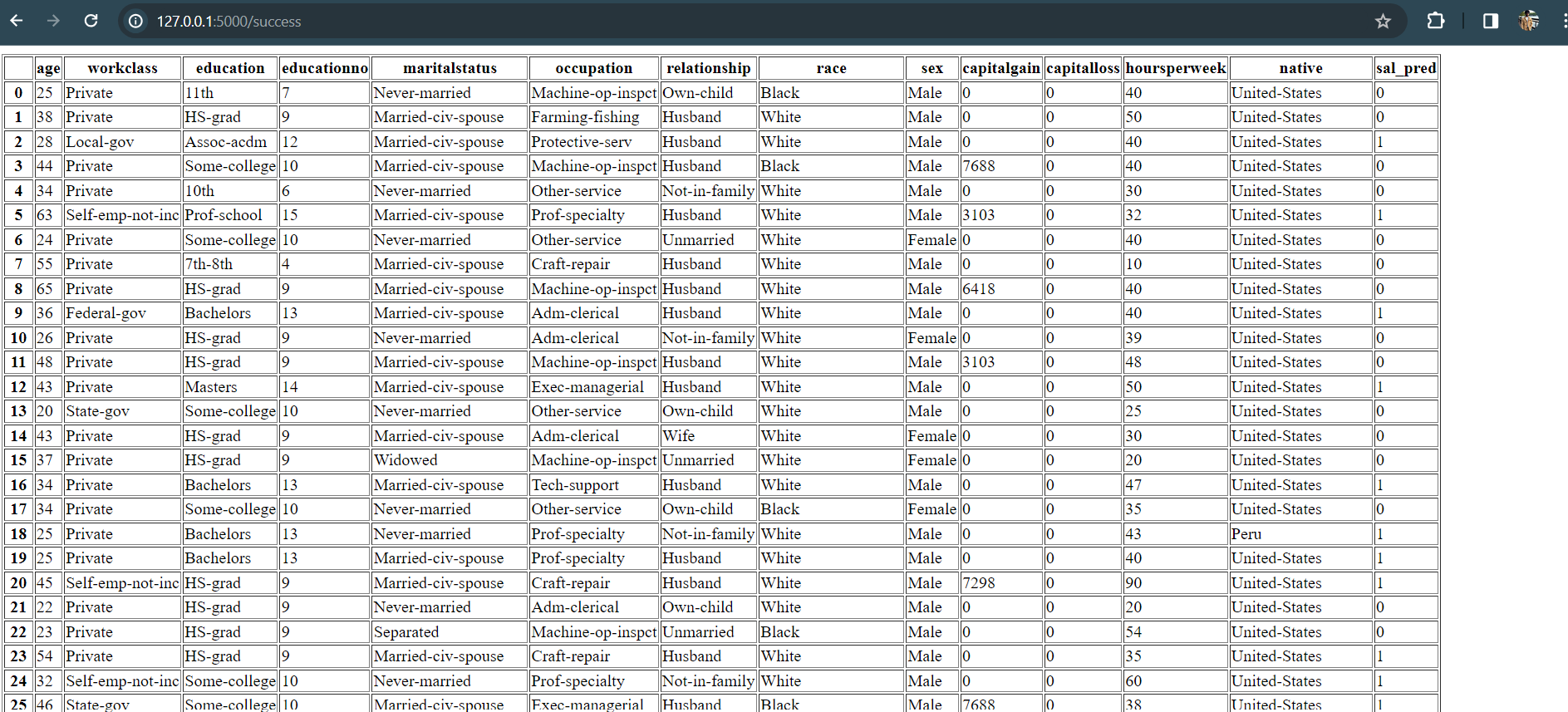
print(NB\_NEW.best\_params\_)

{'alpha': 1.0, 'fit\_prior': True}

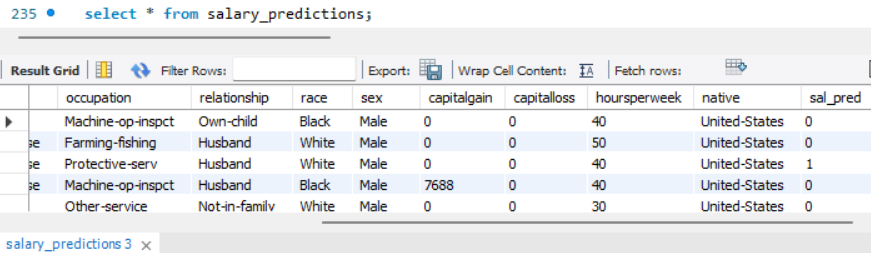
accuracy: 0.78, sensitivity: 0.77, specificity: 0.78, precision: 0.55



**Deployment of Naïve Bayes model**

****

**Saving the predicted values in MySQL for monitoring**

****