# Topic: Support Vector Machines (SVM)

**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: SVM**

**Guidelines:**

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

**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.**

**3. Data Pre-processing**

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

**3.2 Outlier Treatment.**

**4. Exploratory Data Analysis (EDA):**

**4.1 Summary.**

**4.2 Univariate analysis.**

**4.3 Bivariate analysis.**

**5. Model Building**

* 1. **Build the model on the scaled data (try multiple options)**
  2. **Use the SVM algorithm.**
  3. **Train and test the model and compare accuracies by building a confusion matrix and using different hyperparameters.**
  4. **Briefly explain the model output in the documentation.**

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

**Problem Statement: -**

1. A construction firm wants to develop a suburban locality with new infrastructure but they might incur losses if they cannot sell the properties. To overcome this, they consult an analytics firm to get insights into how densely the area is populated and the income levels of residents. Use the Support Vector Machines algorithm on the given dataset and draw out insights and also comment on the viability of investing in that area.



|  |  |  |  |
| --- | --- | --- | --- |
| Name of Feature | 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 | Irrelevant |
| capitalloss | Capital loss | Quantitative | Irrelevant |
| hoursperweek | Number of hours worked per week | Quantitative | Relevant |
| native | Native country | Nominal | Relevant |
| Salary | Salary level | Nominal | Relevant |

**Code:**

'''CRISP-ML(Q)

a. Business & Data Understanding

A construction firm wants to develop a suburban locality with new infrastructure but they might incur losses if they cannot sell the properties. To overcome this, they consult an analytics firm to get insights into how densely the area is populated and the income levels of residents. Use the Support Vector Machines algorithm on the given dataset and draw out insights and also comment on the viability of investing in that area.

i. Business Objective - Maximize viablity of investing

ii. Business Constraint - Minimize Prediction Errors

Success Criteria:

1. Business Success Criteria - Increase finding viable areas by atleast 20%

2. ML Success Criteria - Achieve a prediction accuracy of atleast by 80%

3. Economic Success Criteria - Increase the profit atleast by 15%

Data Collection - Data is collected to find insights into how densely the area is populated and the income levels of residents

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 Irrelevant

capitalloss Capital loss Quantitative Irrelevant

hoursperweek Number of hours worked per week Quantitative Relevant

native Native country Nominal Relevant

Salary Salary level Nominal Relevant

'''

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.compose import ColumnTransformer

from sklearn.pipeline import Pipeline

from sklearn.impute import SimpleImputer

from feature\_engine.outliers import Winsorizer

from sklearn.preprocessing import MinMaxScaler

from sklearn.preprocessing import StandardScaler, OneHotEncoder

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

from sklearn.svm import SVC

from sklearn.model\_selection import RandomizedSearchCV

import pickle, joblib

from sqlalchemy import create\_engine, text

user = 'root' # user name

pw = '1234' # password

db = 'salary\_db' # database name

engine = create\_engine(f"mysql+pymysql://{user}:{pw}@localhost/{db}")

salary = pd.read\_csv(r"C:/Users/Lenovo/Downloads/Study material/Data Science/SVM\_classifier/Assignment/SVM/SalaryData\_Train.csv")

salary.to\_sql('salary\_svm', con = engine, if\_exists = 'replace', chunksize = 1000, index = False)

sql = 'select \* from salary\_svm;'

salary = pd.read\_sql\_query(text(sql), engine.connect()).head(5000)

salary.describe()

salary.drop(columns = ["capitalgain", "capitalloss"], inplace = True)

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

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

# Predictors and Target

X = salary.iloc[:, :11]

Y = salary.iloc[:, 11]

# Define numeric and categorical features

numeric\_features = X.select\_dtypes(exclude=['object']).columns

categorical\_features = X.select\_dtypes(include=['object']).columns

# Outlier Treatment

# Multiple boxplots in a single visualization.

# Columns with larger scales affect other columns.

# Below code ensures each column gets its own y-axis.

X[numeric\_features].plot(kind = 'box', subplots = True, sharey = False, figsize = (15, 8))

'''sharey True or 'all': x- or y-axis will be shared among all subplots.

False or 'none': each subplot x- or y-axis will be independent.'''

# increase spacing between subplots

plt.subplots\_adjust(wspace = 0.75) # ws is the width of the padding between subplots, as a fraction of the average Axes width.

plt.show()

X.info()

# Pipeline for numerical feature preprocessing

numeric\_pipeline = Pipeline(steps=[

('winsor', Winsorizer(capping\_method='iqr', tail='both', fold=1.5, variables= list(numeric\_features))),

('impute', SimpleImputer(strategy='mean')),

('scale', MinMaxScaler())

])

# Pipeline for categorical feature preprocessing

categorical\_pipeline = Pipeline(steps=[

('onehot', OneHotEncoder())

])

# Preprocessor using ColumnTransformer

preprocessor = ColumnTransformer([('numeric', numeric\_pipeline, numeric\_features),

('categorical', categorical\_pipeline, categorical\_features)], remainder='passthrough')

# Fit and transform the preprocessor to the data

preprocessed = preprocessor.fit(X)

# Save the data preprocessing pipeline

joblib.dump(preprocessed, 'preprocessor.pkl')

clean\_data1 = pd.DataFrame(preprocessed.transform(X).toarray(), columns = list(preprocessed.get\_feature\_names\_out()))

clean\_data1.info()

# Boxplot

clean\_data1[["numeric\_\_age", "numeric\_\_educationno", "numeric\_\_hoursperweek"]].plot(kind = 'box', subplots = True, sharey = False, figsize = (15, 8))

plt.subplots\_adjust(wspace = 0.75) # ws is the width of the padding between subplots, as a fraction of the average Axes width.

plt.show()

X2 = clean\_data1

# Data Partition into Train and Test

train\_X, test\_X, train\_y, test\_y = train\_test\_split(X2, Y, test\_size = 0.2, stratify = Y)

# Support Vector Classifier

# SVC with linear kernel trick

model\_linear = SVC(kernel = "linear")

model1 = model\_linear.fit(train\_X, train\_y)

a = model\_linear.coef\_

# print(model\_linear.decision\_function\_shape)

pred\_test\_linear = model\_linear.predict(test\_X)

# Accuracy

np.mean(pred\_test\_linear == test\_y)

### Hyperparameter Optimization

# RandomizedSearchCV

# Base model

model = SVC()

# Parameters set

parameters = {'C': [0.1, 1, 10, 100],

'gamma': [1, 0.1, 0.01, 0.001],

'kernel': ['linear', 'poly', 'rbf', 'sigmoid']}

# Randomized Search Technique for exhaustive search for best model

rand\_search = RandomizedSearchCV(model, parameters, n\_iter = 10,

n\_jobs = 3, cv = 3, scoring = 'accuracy', random\_state = 0)

# Fitting the model for grid search

randomised = rand\_search.fit(train\_X, train\_y)

# Best parameters

randomised.best\_params\_

# Best Model

best = randomised.best\_estimator\_

# Evaluate on Test data

pred\_test = best.predict(test\_X)

np.mean(pred\_test == test\_y)

# Saving the best model - rbf kernel model

pickle.dump(best, open('svc\_rcv.pkl', 'wb'))

########## New Data Prediction ####

data = pd.read\_csv(r"C:/Users/Lenovo/Downloads/Study material/Data Science/SVM\_classifier/Assignment/SVM/test.csv")

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

numeric\_features = data.select\_dtypes(exclude = ['object']).columns

model1 = pickle.load(open('svc\_rcv.pkl', 'rb'))

preprocessed = joblib.load('preprocessor.pkl')

clean = pd.DataFrame(preprocessed.transform(data).toarray(), columns = list(preprocessed.get\_feature\_names\_out()))

prediction = pd.DataFrame(model1.predict(clean), columns = ['sal\_pred'])

final = pd.concat([prediction, data], axis = 1)

**Output:**

X1

Out[43]:

age educationno capitalgain capitalloss hoursperweek

0 39.0 13.0 2174.0 0.0 40.0

1 50.0 13.0 0.0 0.0 13.0

2 38.0 9.0 0.0 0.0 40.0

3 53.0 7.0 0.0 0.0 40.0

4 28.0 13.0 0.0 0.0 40.0

... ... ... ... ...

30156 27.0 12.0 0.0 0.0 38.0

30157 40.0 9.0 0.0 0.0 40.0

30158 58.0 9.0 0.0 0.0 40.0

30159 22.0 9.0 0.0 0.0 20.0

30160 52.0 9.0 15024.0 0.0 40.0

[30161 rows x 5 columns]

X1.isna().sum()

Out[44]:

age 0

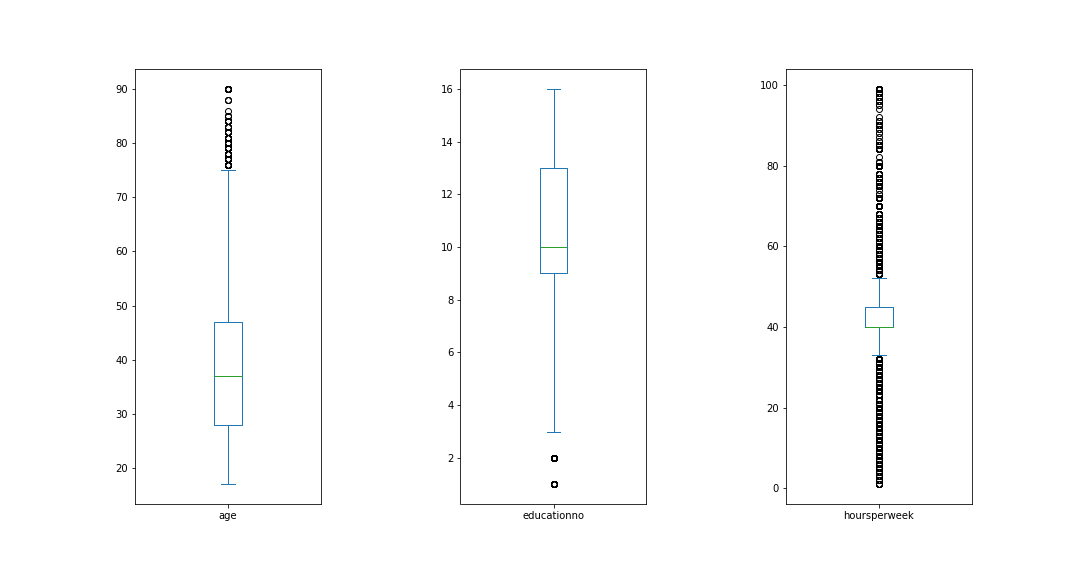
educationno 0

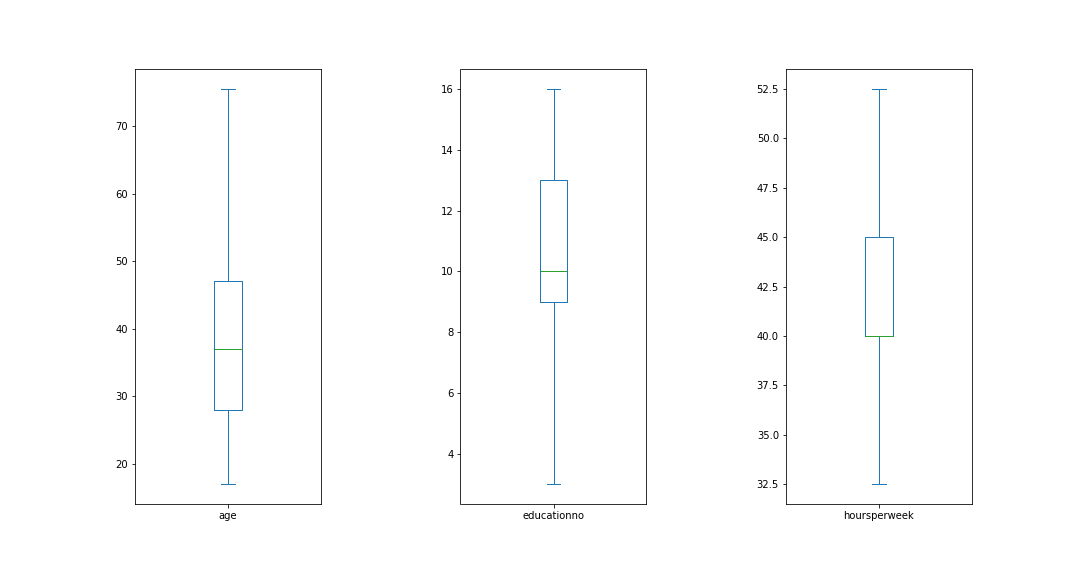
capitalgain 0

capitalloss 0

hoursperweek 0

dtype: int64





X2.describe()

Out[27]:

age educationno hoursperweek

count 30161.000000 30161.000000 30161.000000

mean 0.365891 0.548408 0.445970

std 0.222678 0.194265 0.307302

min 0.000000 0.000000 0.000000

25% 0.188034 0.461538 0.375000

50% 0.341880 0.538462 0.375000

75% 0.512821 0.769231 0.625000

max 1.000000 1.000000 1.000000

X2 = clean\_data1

X2

Out[130]:

numeric\_\_age ... categorical\_\_native\_ Yugoslavia

0 0.376068 ... 0.0

1 0.564103 ... 0.0

2 0.358974 ... 0.0

3 0.615385 ... 0.0

4 0.188034 ... 0.0

... ... ...

30156 0.170940 ... 0.0

30157 0.393162 ... 0.0

30158 0.700855 ... 0.0

30159 0.085470 ... 0.0

30160 0.598291 ... 0.0

[30161 rows x 100 columns]

np.mean(pred\_test\_linear == test\_y)

Out[139]: 0.796

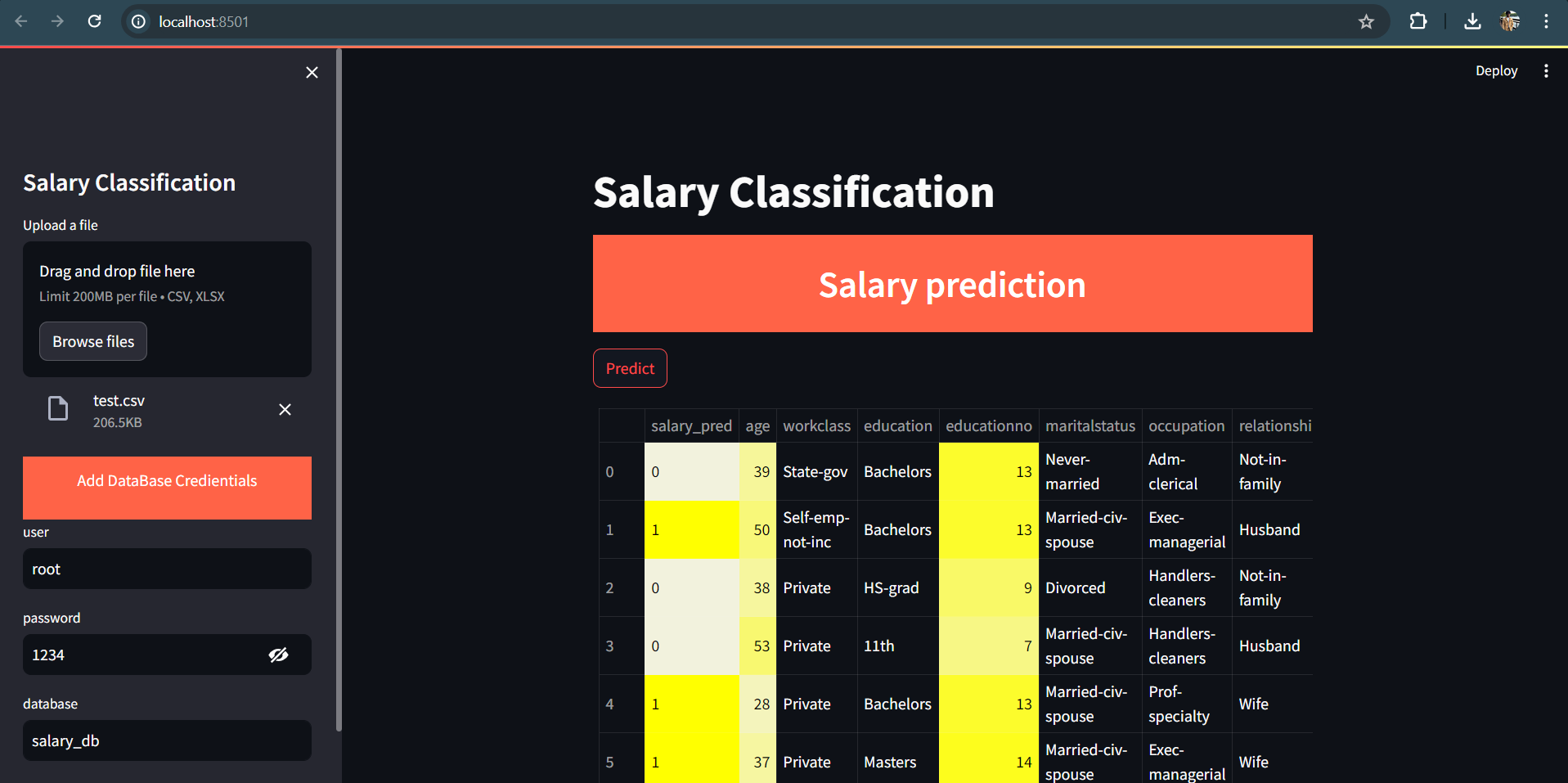
randomised.best\_params\_

Out[16]: {'kernel': 'rbf', 'gamma': 0.01, 'C': 1}

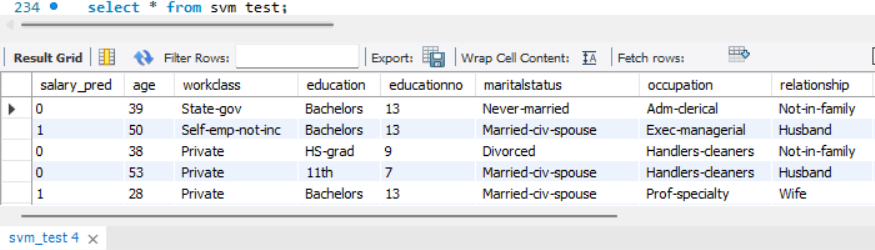
np.mean(pred\_test == test\_y)

Out[21]: 0.827

**Deployment of SVM model using Streamlit**



**Saving Predicted values in MySQL for monitoring**

****