Case Study: Logistic Regression

Income Group Classification Case Study

Problem Statement

An NGO that works with the government on matters of social policy to bring about a change in the lives of underprivileged sections of society. They are given a task of coming up with a policy framework using a dataset that the government received from WHO.

Objective

You, as a Data Scientist , are tasked with analyzing the data provided to identify the different factors that influence the income of an individual, build a prediction model that can help the government formulate policies for the right pockets of the society, and share a proposal for the government.

Dataset

The data contains characteristics of the people

- age: continuous - age of a Person
- workclass: Where does a person works - categorical - (Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked, Unknown)
- fnlwgt: continuous - Weight assigned by Current Population Survey (CPS) - People with similar demographic characteristics should have similar weights since it is a feature aimed to allocate similar weights to people with similar demographic characteristics.
- education: Degree the person has - categorical - (Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool)
- education-num: no. of years a person studied - continuous.
- marital-status: categorical - (Married, Never-married, Not-married)
- occupation: categorical - (Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces, Unknown)
- race: categorical - (White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black)
- sex: categorical - (Female, Male)
- capital-gain: Investment gain of the person other than salary - continuous
- capital-loss: Loss from investments - continuous
- hours-per-week: No. of hours a person works - continuous.
- native-contienent: categorical - (North America, Asia, Other, Europe, South America)
- salary: >50K, <=50K (dependent variable, the salary is in Dollars per year)

Loading Libraries

```
# this will help in making the Python code more structured automatically (good coding practi
#%load_ext nb_black

import warnings

warnings.filterwarnings("ignore")
from statsmodels.tools.sm_exceptions import ConvergenceWarning

warnings.simplefilter("ignore", ConvergenceWarning)

# Libraries to help with reading and manipulating data

import pandas as pd
```

```
import numpy as np

# Library to split data
from sklearn.model_selection import train_test_split

# libaries to help with data visualization
import matplotlib.pyplot as plt
import seaborn as sns

# Removes the limit for the number of displayed columns
pd.set_option("display.max_columns", None)
# Sets the limit for the number of displayed rows
pd.set_option("display.max_rows", 200)


# To build model for prediction
import statsmodels.stats.api as sms
from statsmodels.stats.outliers_influence import variance_inflation_factor
import statsmodels.api as sm
from statsmodels.tools.tools import add_constant
from sklearn.linear_model import LogisticRegression


# To get diferent metric scores
from sklearn.metrics import (
    f1_score,
    accuracy_score,
    recall_score,
    precision_score,
    confusion_matrix,
    roc_auc_score,
    ConfusionMatrixDisplay,
    precision_recall_curve,
    roc_curve,
)
```

Loading data

```
who = pd.read_csv("who_data.csv")


# copying data to another variable to avoid any changes to original data
data = who.copy()
```

Overview of the data

View the first and last 5 rows of the dataset.

```
data.head()
```

	age	workclass	fnlwgt	education	education_no_of_years	marital_status	occupation
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical
1	50	Self-emp-not-inc	83311	Bachelors	13	married	Exec-managerial
2	38	Private	215646	HS-grad	9	not_married	Handlers-cleaners
3	53	Private	234721	11th	7	married	Handlers-cleaners
4	28	Private	338409	Bachelors	13	married	Prof-specialty

```
data.tail()
```

	age	workclass	fnlwgt	education	education_no_of_years	marital_status	occupat
32526	27	Private	257302	Assoc-acdm	12	married	T sup
32527	40	Private	154374	HS-grad	9	married	Mach op-in
32528	58	Private	151910	HS-grad	9	not_married	A cle
32529	22	Private	201490	HS-grad	9	Never-married	A cle
32530	52	Self-emp-inc	287927	HS-grad	9	married	E manag

- The dataset contains information about different types of people.

Understand the shape of the dataset.

```
data.shape
```

```
(32531, 14)
```

- The dataset has 32531 rows and 14 columns

Check the data types of the columns for the dataset.

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32531 entries, 0 to 32530
Data columns (total 14 columns):
 #   Column                  Non-Null Count  Dtype 
---  ------                  --------------  ----- 
 0   age                     32531 non-null  int64 
 1   workclass               32531 non-null  object
 2   fnlwgt                  32531 non-null  int64 
 3   education               32531 non-null  object
 4   education_no_of_years   32531 non-null  int64 
 5   marital_status          32531 non-null  object
 6   occupation              32531 non-null  object
 7   race                    32531 non-null  object
 8   sex                     32531 non-null  object
 9   capital_gain            32531 non-null  int64 
 10  capital_loss            32531 non-null  int64 
 11  working_hours_per_week  32531 non-null  int64 
 12  native_contienent       32531 non-null  object
 13  salary                  32531 non-null  object
dtypes: int64(6), object(8)
memory usage: 3.5+ MB
```

- Dependent variable is the salary which is of categorical type.
- age, fnlwgt, education_no_of_years, capital gain, capital loss and working_hours_per_week are of integer type while other variables are of categorical type.
- There are no missing values in the dataset.

Checking for duplicate values

```
data.duplicated().sum()
```

```
0
```

- There are no duplicate values in the dataset.

Checking for missing values

```
data.isnull().sum()
```

```
age                     0
workclass               0
fnlwgt                  0
education               0
education_no_of_years   0
marital_status          0
occupation              0
race                    0
sex                     0
capital_gain            0
capital_loss            0
working_hours_per_week  0
native_contienent       0
salary                  0
dtype: int64
```

- There are no missing values in the dataset

Statistical summary

```
data.describe()
```

	age	fnlwgt	education_no_of_years	capital_gain	capital_loss	wo
count	32531.000000	3.253100e+04	32531.000000	32531.000000	32531.000000	
mean	38.588362	1.897882e+05	10.081953	1078.642649	87.384341	
std	13.637644	1.055642e+05	2.571842	7388.624210	403.137260	
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	
25%	28.000000	1.178150e+05	9.000000	0.000000	0.000000	
50%	37.000000	1.783700e+05	10.000000	0.000000	0.000000	
75%	48.000000	2.370190e+05	12.000000	0.000000	0.000000	
max	90.000000	1.484705e+06	16.000000	99999.000000	4356.000000	

- `age` : Average age of people in the dataset is around 38 years, age has a wide range from 17 to 90 years.
- `education_no_of_years` : The average number of years spent in education is ~10 years. There's a large difference between the minimum value and 25th percentile which indicates

that there might be outliers present in this variable.

- `capital_gain` : There's a huge difference in the 75th percentile and maximum value of capital_gain indicating the presence of outliers. Also, 75% of the observations are 0.
- `capital_loss` : Same as capital gain there's a huge difference in the 75th percentile and maximum value indicating the presence of outliers. Also, 75% of the observations are 0.
- `working_hours_per_week` : On average people work for 40 hours a week. A vast difference in minimum value and 25th percentile, as well as 75th percentile and the maximum value, indicates that there might be outliers present in the variable.

Actions for data pre-processing:

- Many variables have outliers that need to be treated.
- We can drop capital_gain and capital_loss as most of the observations are 0 in these variables, and if the government already knows a person's capital gain or loss it will be easier for them to classify a person into a salary group.

Data Preprocessing

- We can drop capital_gain and capital_loss as most of the observations are 0 in these variables, and if the government already knows a person's capital gain or loss it will be easier for them to classify a person into a salary group
- There are many outliers in the data which we will treat (perform capping of outliers)
    - All the values smaller than the lower whisker will be assigned the value of the lower whisker, and all the values above the upper whisker will be assigned the value of the upper whisker.

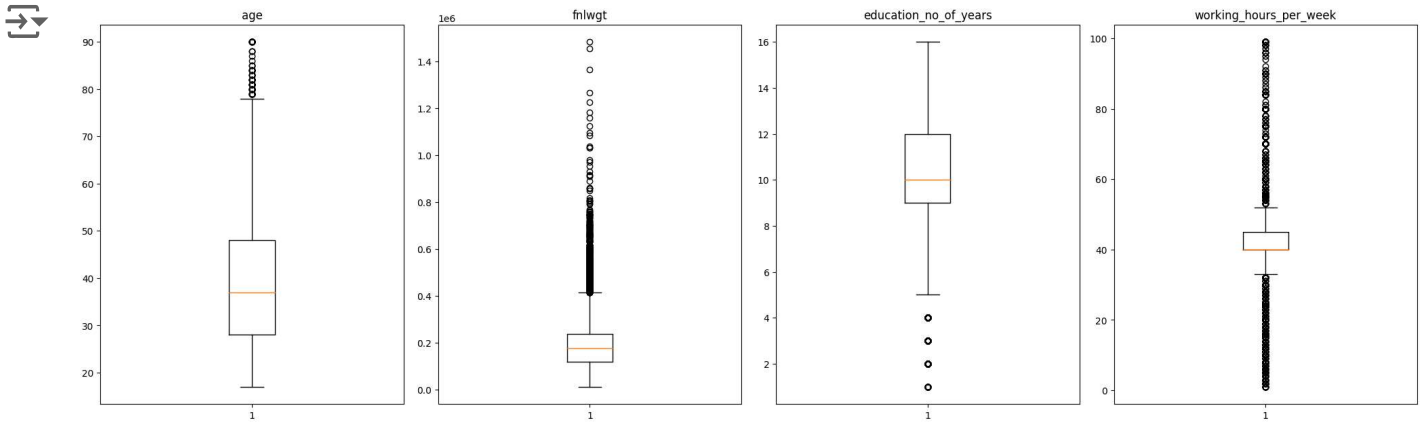
Dropping capital_gain and capital_loss

```
data.drop(["capital_gain", "capital_loss"], axis=1, inplace=True)
```

Outlier Detection

```
numerical_col = data.select_dtypes(include=np.number).columns.tolist()
plt.figure(figsize=(20, 30))

for i, variable in enumerate(numerical_col):
    plt.subplot(5, 4, i + 1)
```

```
    plt.boxplot(data[variable], whis=1.5)
    plt.tight_layout()
    plt.title(variable)

plt.show()
```

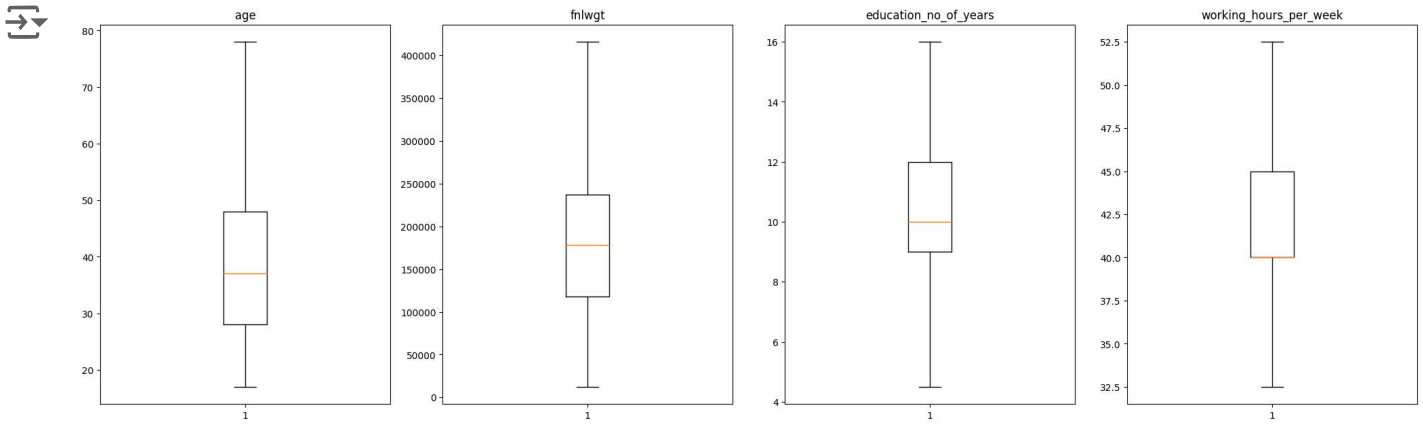


- Age and fnlwgt have upper outliers.
- education_no_years has lower outliers.
- working_hours_per_week has both upper and lower outliers.

Outlier Treatment

```
# functions to treat outliers by flooring and capping


def treat_outliers(df, col):
    """
    Treats outliers in a variable

    df: dataframe
    col: dataframe column
    """
    Q1 = df[col].quantile(0.25)  # 25th quantile
    Q3 = df[col].quantile(0.75)  # 75th quantile
    IQR = Q3 - Q1
    Lower_Whisker = Q1 - 1.5 * IQR
    Upper_Whisker = Q3 + 1.5 * IQR
```

```
        # all the values smaller than Lower_Whisker will be assigned the value of Lower_Whisker
        # all the values greater than Upper_Whisker will be assigned the value of Upper_Whisker
        df[col] = np.clip(df[col], Lower_Whisker, Upper_Whisker)

        return df


    def treat_outliers_all(df, col_list):
        """
        Treat outliers in a list of variables

        df: dataframe
        col_list: list of dataframe columns
        """
        for c in col_list:
            df = treat_outliers(df, c)

        return df


    numerical_col = data.select_dtypes(include=np.number).columns.tolist()
    data = treat_outliers_all(data, numerical_col)


    # let's look at box plot to see if outliers have been treated or not
    plt.figure(figsize=(20, 30))

    for i, variable in enumerate(numerical_col):
        plt.subplot(5, 4, i + 1)
        plt.boxplot(data[variable], whis=1.5)
        plt.tight_layout()
        plt.title(variable)

    plt.show()
```



Data Preparation for Modeling

Encoding >50K as 0 and <=50K as 1 as government wants to find underprivileged section of society.

```
data["salary"] = data["salary"].apply(lambda x: 1 if x == " <=50K" else 0)
```

Creating training and test sets.

```
# specifying the independent  and dependent variables
X = data.drop(["salary"], axis=1)
Y = data["salary"]

# adding a constant to the independent variables
X = sm.add_constant(X)

# creating dummy variables
X = pd.get_dummies(X, drop_first=True)

#Converting values to float
X = X.astype(float)

# splitting data in train and test sets
X_train, X_test, y_train, y_test = train_test_split(
    X, Y, test_size=0.30, random_state=1
)
```

Model Building - Logistic Regression

- We will now perform logistic regression using statsmodels, a Python module that provides functions for the estimation of many statistical models, as well as for conducting statistical tests, and statistical data exploration.
- Using statsmodels, we will be able to check the statistical validity of our model - identify the significant predictors from p-values that we get for each predictor variable.

```
# fitting logistic regression model
logit = sm.Logit(y_train, X_train.astype(float))
lg = logit.fit(disp=False)

print(lg.summary())
```

```
==============================================================================
                                 coef    std err          z      P>|z|      [0.
------------------------------------------------------------------------------
const                          9.5200        nan        nan        nan         
age                           -0.0314      0.002    -17.126      0.000      -0.
fnlwgt                     -6.146e-07   2.13e-07     -2.881      0.004   -1.03e
```

education_11th	0.1667	nan	nan	nan	
education_12th	-0.0044	nan	nan	nan	
education_1st-4th	1.3124	nan	nan	nan	
education_5th-6th	0.2102	nan	nan	nan	
education_7th-8th	0.3224	nan	nan	nan	
education_9th	0.4116	nan	nan	nan	
education_Assoc-acdm	-0.0897	nan	nan	nan	
education_Assoc-voc	-0.3665	nan	nan	nan	
education_Bachelors	-0.5915	nan	nan	nan	
education_Doctorate	-0.9450	nan	nan	nan	
education_HS-grad	-0.1775	nan	nan	nan	
education_Masters	-0.7981	nan	nan	nan	
education_Preschool	12.0908	nan	nan	nan	
education_Prof-school	-1.3568	nan	nan	nan	
education_Some-college	-0.2672	nan	nan	nan	
marital_status_married	-2.5742	0.069	-37.060	0.000	-2.
marital_status_not_married	-0.4045	0.084	-4.806	0.000	-0.
occupation_Armed-Forces	0.5322	1.529	0.348	0.728	-2.
occupation_Craft-repair	0.0011	0.088	0.013	0.990	-0.
occupation_Exec-managerial	-0.6952	0.085	-8.199	0.000	-0.
occupation_Farming-fishing	1.0593	0.157	6.763	0.000	0.
occupation_Handlers-cleaners	0.8354	0.162	5.169	0.000	0.
occupation_Machine-op-inspct	0.4118	0.115	3.573	0.000	0.
occupation_Other-service	0.9811	0.135	7.294	0.000	0.
occupation_Priv-house-serv	2.3843	1.091	2.185	0.029	0.
occupation_Prof-specialty	-0.4583	0.089	-5.143	0.000	-0.
occupation_Protective-serv	-0.4979	0.143	-3.475	0.001	-0.
occupation_Sales	-0.1410	0.091	-1.555	0.120	-0.
occupation_Tech-support	-0.5943	0.123	-4.815	0.000	-0.
occupation_Transport-moving	0.1759	0.112	1.572	0.116	-0.
occupation_Unknown	8.2720	7.88e+06	1.05e-06	1.000	-1.54e-
race_Asian-Pac-Islander	-0.2499	0.287	-0.871	0.384	-0.
race_Black	-0.2403	0.251	-0.958	0.338	-0.
race_Other	0.4680	0.401	1.167	0.243	-0.
race_White	-0.3346	0.238	-1.403	0.161	-0.
sex_Male	-0.0661	0.059	-1.129	0.259	-0.
native_contienent_europe	-0.2102	0.241	-0.871	0.384	-0.
native_contienent_north_america	-0.2393	0.191	-1.251	0.211	-0.
native_contienent_other	0.2389	0.218	1.097	0.272	-0.
native_contienent_south_america	1.0479	0.526	1.992	0.046	0.

==========

Observations

- Negative values of the coefficient show that the probability of a person having a salary less than 50K decreases with the increase of the corresponding attribute value.
- Positive values of the coefficient show that the probability of a person having a salary less than 50K increases with the increase of the corresponding attribute value.
- p-value of a variable indicates if the variable is significant or not. If we consider the significance level to be 0.05 (5%), then any variable with a p-value less than 0.05 would be

considered significant.

Model Performance Evaluation

Model can make wrong predictions as:

1. Predicting a person has a salary <=50K but in reality the salary of the person is >50K.
2. Predicting a person doesn't have a salary <=50K but in reality the salary of the person is <=50k.

Which case is more important?

- Both the cases are important as:
  - If we predict a person has a salary <=50K but actually the salary is >50K then a wrong person will be getting the benefits of the scheme and government might lose resources.
  - If we predict a person doesn't have a salary <=50K but actually the salary is <=50K that person will not be able to avail the benefits of the scheme constituting to opportunity loss.

How to reduce this loss?

- We need to reduce both False Negatives and False Positives
- `f1_score` should be maximized as the greater the f1_score, the higher the chances of reducing both False Negatives and False Positives and identifying both the classes correctly
  - fi_score is computed as

$$f1\_score = \frac{2 * Precision * Recall}{Precision + Recall}$$

First, let's create functions to calculate different metrics and confusion matrix so that we don't have to use the same code repeatedly for each model.

- The model_performance_classification_statsmodels function will be used to check the model performance of models.
- The confusion_matrix_statsmodels function will be used to plot confusion matrix.

```
# defining a function to compute different metrics to check performance of a classification
def model_performance_classification_statsmodels(
    model, predictors, target, threshold=0.5
):
    """
    Function to compute different metrics to check classification model performance
```

```
    model: classifier
    predictors: independent variables
    target: dependent variable
    threshold: threshold for classifying the observation as class 1
    """

    # checking which probabilities are greater than threshold
    pred_temp = model.predict(predictors) > threshold
    # rounding off the above values to get classes
    pred = np.round(pred_temp)

    acc = accuracy_score(target, pred)  # to compute Accuracy
    recall = recall_score(target, pred)  # to compute Recall
    precision = precision_score(target, pred)  # to compute Precision
    f1 = f1_score(target, pred)  # to compute F1-score

    # creating a dataframe of metrics
    df_perf = pd.DataFrame(
        {"Accuracy": acc, "Recall": recall, "Precision": precision, "F1": f1,},
        index=[0],
    )

    return df_perf


print("Training performance:")
model_performance_classification_statsmodels(lg, X_train, y_train)
```

```
Training performance:
```

	Accuracy	Recall	Precision	F1
0	0.835624	0.923759	0.867897	0.894957

Observations

- The f1_score of the model is ~0.89 and we will try to maximize it further
- The variables used to build the model might contain multicollinearity, which will affect the p-values
- We will have to remove multicollinearity from the data to get reliable coefficients and p-values

Detecting and Dealing with Multicollinearity

There are different ways of detecting (or testing for) multicollinearity. One such way is using the Variation Inflation Factor (VIF).

- **Variance Inflation factor**: Variance inflation factors measure the inflation in the variances of the regression coefficients estimates due to collinearities that exist among the predictors. It is a measure of how much the variance of the estimated regression coefficient β_k is "inflated" by the existence of correlation among the predictor variables in the model.
- **General Rule of thumb**:
  - If VIF is 1 then there is no correlation among the kth predictor and the remaining predictor variables, and hence the variance of β_k is not inflated at all
  - If VIF exceeds 5, we say there is moderate multicollinearity
  - If VIF is equal or exceeding 10, it shows signs of high multi-collinearity
- The purpose of the analysis should dictate which threshold to use

```
vif_series = pd.Series(
    [variance_inflation_factor(X_train.values, i) for i in range(X_train.shape[1])],
    index=X_train.columns,
    dtype=float,
)
print("Series before feature selection: \n\n{}\n".format(vif_series))
```

```
Series before feature selection:

const                           0.000000
age                             1.531401
fnlwgt                          1.037491
education_no_of_years                inf
working_hours_per_week          1.237162
workclass_Local-gov             3.041470
workclass_Never-worked               inf
workclass_Private               7.717533
workclass_Self-emp-inc          2.217152
workclass_Self-emp-not-inc      3.487046
workclass_State-gov             2.317984
workclass_Unknown                    inf
workclass_Without-pay           1.015562
education_11th                       inf
education_12th                       inf
education_1st-4th                    inf
education_5th-6th                    inf
education_7th-8th                    inf
education_9th                        inf
education_Assoc-acdm                 inf
education_Assoc-voc                  inf
education_Bachelors                  inf
education_Doctorate                  inf
education_HS-grad                    inf
education_Masters                    inf
education_Preschool                  inf
education_Prof-school                inf
education_Some-college               inf
marital_status_married          1.971883
```

```
marital_status_not_married          1.714786
occupation_Armed-Forces             1.010066
occupation_Craft-repair             2.133802
occupation_Exec-managerial          2.046469
occupation_Farming-fishing          1.401571
occupation_Handlers-cleaners        1.440475
occupation_Machine-op-inspct        1.577192
occupation_Other-service            1.750404
occupation_Priv-house-serv          1.054692
occupation_Prof-specialty           2.317960
occupation_Protective-serv          1.253107
occupation_Sales                    1.891884
occupation_Tech-support             1.239360
occupation_Transport-moving         1.509398
occupation_Unknown                       inf
race_Asian-Pac-Islander             5.305020
race_Black                         10.196674
race_Other                          1.878501
race_White                         13.494644
sex_Male                            1.520198
native_contienent_europe            2.424319
native_contienent_north_america     5.250971
native_contienent_other             2.331141
native_contienent_south_america     1.328881
dtype: float64
```

- `education_no_of_years`, `education`, and some categorical levels of `workclass`, `native_country`, and `race` exhibit high multicollinearity
- We see that some variables have *inf* values for VIF
  - The *inf* values for VIF indicate perfect correlation between variables
- As we saw in our earlier analysis that wherever workclass has unknown values occupation is also unknown.
  - The high correlation between these variables has been highlighted in the VIF values as well
- We will drop `occupation_Unknown` as we get the same information from `workclass_Unknown`

Removing `occupation_Unknown`

```
X_train1 = X_train.drop("occupation_Unknown", axis=1)

vif_series2 = pd.Series(
    [variance_inflation_factor(X_train1.values, i) for i in range(X_train1.shape[1])],
    index=X_train1.columns,
)
print("Series before feature selection: \n\n{}\n".format(vif_series2))
```

```
Series before feature selection:

const                                0.000000
age                                  1.531401
fnlwgt                               1.037491
education_no_of_years                     inf
working_hours_per_week               1.237162
workclass_Local-gov                  3.041470
workclass_Never-worked               1.012785
workclass_Private                    7.717533
workclass_Self-emp-inc               2.217152
workclass_Self-emp-not-inc           3.487046
workclass_State-gov                  2.317984
workclass_Unknown                    3.039325
workclass_Without-pay                1.015562
education_11th                            inf
education_12th                            inf
education_1st-4th                         inf
education_5th-6th                         inf
education_7th-8th                         inf
education_9th                             inf
education_Assoc-acdm                      inf
education_Assoc-voc                       inf
education_Bachelors                       inf
education_Doctorate                       inf
education_HS-grad                         inf
education_Masters                         inf
education_Preschool                       inf
education_Prof-school                     inf
education_Some-college                    inf
marital_status_married               1.971883
marital_status_not_married           1.714786
occupation_Armed-Forces              1.010066
occupation_Craft-repair              2.133802
occupation_Exec-managerial           2.046469
occupation_Farming-fishing           1.401571
occupation_Handlers-cleaners         1.440475
occupation_Machine-op-inspct         1.577192
occupation_Other-service             1.750404
occupation_Priv-house-serv           1.054692
occupation_Prof-specialty            2.317960
occupation_Protective-serv           1.253107
occupation_Sales                     1.891884
occupation_Tech-support              1.239360
occupation_Transport-moving          1.509398
race_Asian-Pac-Islander              5.305020
race_Black                          10.196674
```

```
race_Other                        1.878501
race_White                       13.494644
sex_Male                          1.520198
native_contienent_europe          2.424319
native_contienent_north_america   5.250971
native_contienent_other           2.331141
native_contienent_south_america   1.328881
dtype: float64
```

- `education_no_of_years` and `education` also capture similar information and hence, it is understandable that they have high VIF values
- We will be dropping `education_no_of_years` and `education` one at a time, building separate models, and checking their performances to see which variable has a significant impact on the model's performance

Dropping `education_no_of_years`

```
X_train2 = X_train1.drop("education_no_of_years", axis=1)
vif_series3 = pd.Series(
    [variance_inflation_factor(X_train2.values, i) for i in range(X_train2.shape[1])],
    index=X_train2.columns,
)
print("Series before feature selection: \n\n{}\n".format(vif_series3))
```

```
Series before feature selection:

const                          326.271888
age                              1.531401
fnlwgt                           1.037491
working_hours_per_week           1.237162
workclass_Local-gov              3.041470
workclass_Never-worked           1.012785
workclass_Private                7.717533
workclass_Self-emp-inc           2.217152
workclass_Self-emp-not-inc       3.487046
workclass_State-gov              2.317984
workclass_Unknown                3.039325
workclass_Without-pay            1.015562
education_11th                   2.147415
education_12th                   1.422410
education_1st-4th                1.188352
education_5th-6th                1.338683
education_7th-8th                1.655272
education_9th                    1.524978
education_Assoc-acdm             2.105382
education_Assoc-voc              2.374490
education_Bachelors              5.939193
education_Doctorate              1.564790
education_HS-grad                8.112679
```

```
education_Masters                    2.978449
education_Preschool                  1.064098
education_Prof-school                1.811978
education_Some-college               6.857574
marital_status_married               1.971883
marital_status_not_married           1.714786
occupation_Armed-Forces              1.010066
occupation_Craft-repair              2.133802
occupation_Exec-managerial           2.046469
occupation_Farming-fishing           1.401571
occupation_Handlers-cleaners         1.440475
occupation_Machine-op-inspct         1.577192
occupation_Other-service             1.750404
occupation_Priv-house-serv           1.054692
occupation_Prof-specialty            2.317960
occupation_Protective-serv           1.253107
occupation_Sales                     1.891884
occupation_Tech-support              1.239360
occupation_Transport-moving          1.509398
race_Asian-Pac-Islander              5.305020
race_Black                          10.196674
race_Other                           1.878501
race_White                          13.494644
sex_Male                             1.520198
native_contienent_europe             2.424319
native_contienent_north_america      5.250971
native_contienent_other              2.331141
native_contienent_south_america      1.328881
dtype: float64
```

- Dropping education_no_of_years fixes the multicollinearity in education column.

```
logit2 = sm.Logit(y_train, X_train2.astype(float))
lg2 = logit2.fit(disp=False)

print("Training performance:")
model_performance_classification_statsmodels(lg2, X_train2, y_train)
```

```
Training performance:
```

	Accuracy	Recall	Precision	F1
0	0.835624	0.923759	0.867897	0.894957

- No significant change in the model performance.

Observations:

1. Dropping education_no_of_years, race_ White doesn't have a significant impact on the model performance.

2. We can choose any model to proceed to the next steps.
3. Some of the categorical levels of a variable have VIF>5 which can simply be ignored.

```
print(lg2.summary())
```

```
                           Logit Regression Results
==============================================================================
Dep. Variable:                 salary   No. Observations:                22771
Model:                          Logit   Df Residuals:                    22720
Method:                           MLE   Df Model:                           50
Date:                Thu, 05 Sep 2024   Pseudo R-squ.:                 0.3678
Time:                        13:05:33   Log-Likelihood:                -7965.8
converged:                      False   LL-Null:                       -12600.
Covariance Type:            nonrobust   LLR p-value:                     0.000
================================================================================
                                   coef    std err          z      P>|z|      [0.
--------------------------------------------------------------------------------
const                            8.3433      0.405     20.584      0.000       7.
age                             -0.0314      0.002    -17.126      0.000      -0.
fnlwgt                       -6.146e-07   2.13e-07     -2.881      0.004   -1.03e
working_hours_per_week          -0.0655      0.004    -18.686      0.000      -0.
workclass_Local-gov              0.7669      0.127      6.061      0.000       0.
workclass_Never-worked          16.5407   5517.857      0.003      0.998   -1.08e-
workclass_Private                0.5196      0.105      4.972      0.000       0.
workclass_Self-emp-inc           0.2891      0.137      2.105      0.035       0.
workclass_Self-emp-not-inc       0.9751      0.123      7.899      0.000       0.
workclass_State-gov              0.9400      0.141      6.667      0.000       0.
workclass_Unknown                1.2653      0.156      8.118      0.000       0.
workclass_Without-pay           16.5958   1640.005      0.010      0.992   -3197.
education_11th                  -0.0294      0.239     -0.123      0.902      -0.
education_12th                  -0.3966      0.293     -1.352      0.176      -0.
education_1st-4th                1.6066      0.742      2.166      0.030       0.
education_5th-6th                0.5044      0.364      1.385      0.166      -0.
education_7th-8th                0.6165      0.265      2.325      0.020       0.
education_9th                    0.6077      0.310      1.963      0.050       0.
education_Assoc-acdm            -1.2664      0.199     -6.376      0.000      -1.
education_Assoc-voc             -1.3471      0.190     -7.088      0.000      -1.
education_Bachelors             -1.9643      0.176    -11.133      0.000      -2.
education_Doctorate             -2.9061      0.238    -12.200      0.000      -3.
education_HS-grad               -0.7659      0.172     -4.463      0.000      -1.
education_Masters               -2.3671      0.188    -12.583      0.000      -2.
education_Preschool             16.5732   2562.495      0.006      0.995   -5005.
education_Prof-school           -3.1219      0.224    -13.961      0.000      -3.
education_Some-college          -1.0517      0.174     -6.035      0.000      -1.
marital_status_married          -2.5742      0.069    -37.060      0.000      -2.
marital_status_not_married      -0.4045      0.084     -4.806      0.000      -0.
occupation_Armed-Forces          0.5322      1.529      0.348      0.728      -2.
occupation_Craft-repair          0.0011      0.088      0.013      0.990      -0.
occupation_Exec-managerial      -0.6952      0.085     -8.199      0.000      -0.
occupation_Farming-fishing       1.0593      0.157      6.763      0.000       0.
occupation_Handlers-cleaners     0.8354      0.162      5.169      0.000       0.
occupation_Machine-op-inspct     0.4118      0.115      3.573      0.000       0.
occupation_Other-service         0.9811      0.135      7.294      0.000       0.
occupation_Priv-house-serv       2.3843      1.091      2.185      0.029       0.
```

occupation_Prof-specialty	-0.4583	0.089	-5.143	0.000	-0.6
occupation_Protective-serv	-0.4979	0.143	-3.475	0.001	-0.7
occupation_Sales	-0.1410	0.091	-1.555	0.120	-0.3
occupation_Tech-support	-0.5943	0.123	-4.815	0.000	-0.8
occupation_Transport-moving	0.1759	0.112	1.572	0.116	-0.6
race_Asian-Pac-Islander	-0.2499	0.287	-0.871	0.384	-0.8
race_Black	-0.2403	0.251	-0.958	0.338	-0.7

Removing high p-value variables

- For other attributes present in the data, the p-values are high only for few dummy variables and since only one (or some) of the categorical levels have a high p-value we will drop them iteratively as sometimes p-values change after dropping a variable. So, we'll not drop all variables at once.
- Instead, we will do the following repeatedly using a loop:
  - Build a model, check the p-values of the variables, and drop the column with the highest p-value.
  - Create a new model without the dropped feature, check the p-values of the variables, and drop the column with the highest p-value.
  - Repeat the above two steps till there are no columns with p-value > 0.05.

Note: The above process can also be done manually by picking one variable at a time that has a high p-value, dropping it, and building a model again. But that might be a little tedious and using a loop will be more efficient.

```
# initial list of columns
cols = X_train2.columns.tolist()

# setting an initial max p-value
max_p_value = 1

while len(cols) > 0:
    # defining the train set
    X_train_aux = X_train2[cols]

    # fitting the model
    model = sm.Logit(y_train, X_train_aux).fit(disp=False)

    # getting the p-values and the maximum p-value
    p_values = model.pvalues
    max_p_value = max(p_values)

    # name of the variable with maximum p-value
```

```
        feature_with_p_max = p_values.idxmax()

        if max_p_value > 0.05:
            cols.remove(feature_with_p_max)
        else:
            break

selected_features = cols
print(selected_features)
```

```
['const', 'age', 'fnlwgt', 'working_hours_per_week', 'workclass_Local-gov', 'workclass_P
```

```
X_train3 = X_train2[selected_features]


logit3 = sm.Logit(y_train, X_train3.astype(float))
lg3 = logit3.fit(disp=False)

print(lg3.summary())
```

```
                           Logit Regression Results
==============================================================================
Dep. Variable:                 salary   No. Observations:                22771
Model:                          Logit   Df Residuals:                    22735
Method:                           MLE   Df Model:                           35
Date:                Thu, 05 Sep 2024   Pseudo R-squ.:                  0.3666
Time:                        13:05:57   Log-Likelihood:                -7980.7
converged:                       True   LL-Null:                       -12600.
Covariance Type:            nonrobust   LLR p-value:                     0.000
================================================================================================
                                     coef    std err          z      P>|z|      [0.025
------------------------------------------------------------------------------------------------
const                              8.0557      0.233     34.544      0.000       7.599
age                               -0.0314      0.002    -17.178      0.000      -0.035
fnlwgt                         -6.273e-07   2.12e-07     -2.964      0.003   -1.04e-06
working_hours_per_week            -0.0650      0.003    -19.014      0.000      -0.072
workclass_Local-gov                0.6101      0.103      5.930      0.000       0.408
workclass_Private                  0.3585      0.072      4.976      0.000       0.217
workclass_Self-emp-not-inc         0.8012      0.094      8.485      0.000       0.616
workclass_State-gov                0.7793      0.121      6.466      0.000       0.543
workclass_Unknown                  1.0881      0.132      8.229      0.000       0.829
education_1st-4th                  1.5989      0.729      2.192      0.028       0.169
education_7th-8th                  0.6164      0.229      2.687      0.007       0.167
education_9th                      0.5931      0.279      2.123      0.034       0.046
education_Assoc-acdm              -1.2818      0.146     -8.791      0.000      -1.568
education_Assoc-voc               -1.3671      0.134    -10.168      0.000      -1.631
education_Bachelors               -1.9848      0.114    -17.449      0.000      -2.208
education_Doctorate               -2.9087      0.196    -14.845      0.000      -3.293
education_HS-grad                 -0.7808      0.108     -7.260      0.000      -0.992
education_Masters                 -2.3806      0.131    -18.149      0.000      -2.638
education_Prof-school             -3.1388      0.178    -17.616      0.000      -3.488
education_Some-college            -1.0739      0.111     -9.666      0.000      -1.292
```

```
marital_status_married            -2.5792      0.067    -38.449      0.000      -2.711
marital_status_not_married        -0.3935      0.084     -4.711      0.000      -0.557
occupation_Exec-managerial        -0.7135      0.063    -11.324      0.000      -0.837
occupation_Farming-fishing         1.0603      0.142      7.450      0.000       0.781
occupation_Handlers-cleaners       0.7954      0.149      5.340      0.000       0.503
occupation_Machine-op-inspct       0.3845      0.097      3.967      0.000       0.195
occupation_Other-service           0.9771      0.122      8.026      0.000       0.738
occupation_Priv-house-serv         2.4240      1.082      2.240      0.025       0.303
occupation_Prof-specialty         -0.4775      0.072     -6.675      0.000      -0.618
occupation_Protective-serv        -0.5487      0.130     -4.205      0.000      -0.804
occupation_Sales                  -0.1563      0.068     -2.299      0.021      -0.290
occupation_Tech-support           -0.6295      0.110     -5.700      0.000      -0.846
race_Other                         0.6816      0.329      2.069      0.039       0.036
race_White                        -0.1545      0.067     -2.301      0.021      -0.286
native_contienent_other            0.4532      0.149      3.041      0.002       0.161
native_contienent_south_america    1.2598      0.489      2.576      0.010       0.301
==============================================================================
```

Now no categorical feature has p-value greater than 0.05, so we'll consider the features in *X_train3* as the final ones and *lg3* as final model.

Coefficient Interpretations

- Coefficient of some levels of education, workclass, and native country are positive an increase in these will lead to increase in chances of a person having <=50K salary.
- Coefficient of age, fnlwgt, marital_status, working_hours_per_week, some levels of education, workclass, and native country are negative increase in these will lead to decrease in chances of a person having <=50K salary.

Converting coefficients to odds

- The coefficients (βs) of the logistic regression model are in terms of $log(odds)$ and to find the odds, we have to take the exponential of the coefficients
- Therefore, $odds = exp(\beta)$
- The percentage change in odds is given as $(exp(\beta) - 1) * 100$

```
# converting coefficients to odds
odds = np.exp(lg3.params)

# finding the percentage change
perc_change_odds = (np.exp(lg3.params) - 1) * 100

# removing limit from number of columns to display
```

```
pd.set_option("display.max_columns", None)

# adding the odds to a dataframe
pd.DataFrame({"Odds": odds, "Change_odd%": perc_change_odds}, index=X_train3.columns).T
```

	const	age	fnlwgt	working_hours_per_week	workclass_Loca g
Odds	3151.769230	0.969130	0.999999	0.937106	1.8406
Change_odd%	315076.922967	-3.086978	-0.000063	-6.289395	84.0620

Coefficient interpretations

- `age` : Holding all other features constant a 1 unit change in Age will decrease the odds of a person having salary <=50k by ~0.97 times or a ~3% decrease in odds of having salary <=50K.
- `working_hours_per_week` : Holding all other features constant a 1 unit change in the working_hours_per_week will decrease the odds of a person having salary <=50k by ~0.94 times or a decrease of ~6.3% decrease in odds of having salary <=50K.
- `marital_status` : The odds of a person whose marital status is married having a salary <=50K is ~0.076 less than the person who was never married or ~92.4% fewer odds of having salary <=50K than the person who was never married. Similarly, the odds of a person whose marital status is not_married having a salary <=50K is ~0.67 times less than the person who was never married or ~32.5% less odds of having a salary <=50K than the person who was never married. [the dropped category Never_married is taken as a reference level]
- `education` : The odds of a person who is HS-grad having salary <=50K is ~0.45 times less than the person who has done education 1st-9th, 11th or Preschool or ~54.1% fewer odds of having salary <=50K. Similarly, The odds of a person who has a Doctorate having a salary <=50K is 0.05 times less than the person who has done education 1st-9th or Preschool or ~95.00% less odds of having a salary <=50K. [the dropped categories education_ 1st-4th,education_ 5th-6th, education_ 7th-8th, education_ 9th, education_ 11th, education_ Preschool are taken as a reference level]

Interpretation for other attributes can be done similarly.

Checking performance of the new model

Training set performance

```
log_reg_model_train_perf = model_performance_classification_statsmodels(
    lg3, X_train3, y_train
```

```
)

print("Training performance:")
log_reg_model_train_perf
```

Training performance:

	Accuracy	Recall	Precision	F1
0	0.835536	0.924048	0.867602	0.894936

Test set performance

- We have to first drop the columns from the test set that were dropped from the training set.

```
X_test3 = X_test[list(X_train3.columns)]


log_reg_model_test_perf = model_performance_classification_statsmodels(
    lg3, X_test3, y_test
)

print("Test performance:")
log_reg_model_test_perf
```

Test performance:

	Accuracy	Recall	Precision	F1
0	0.830635	0.917239	0.867838	0.891855

- The model is giving a good f1_score of ~0.895 and ~0.892 on the train and test sets respectively
- As the train and test performances are comparable, the model is not overfitting
- Moving forward we will try to improve the performance of the model

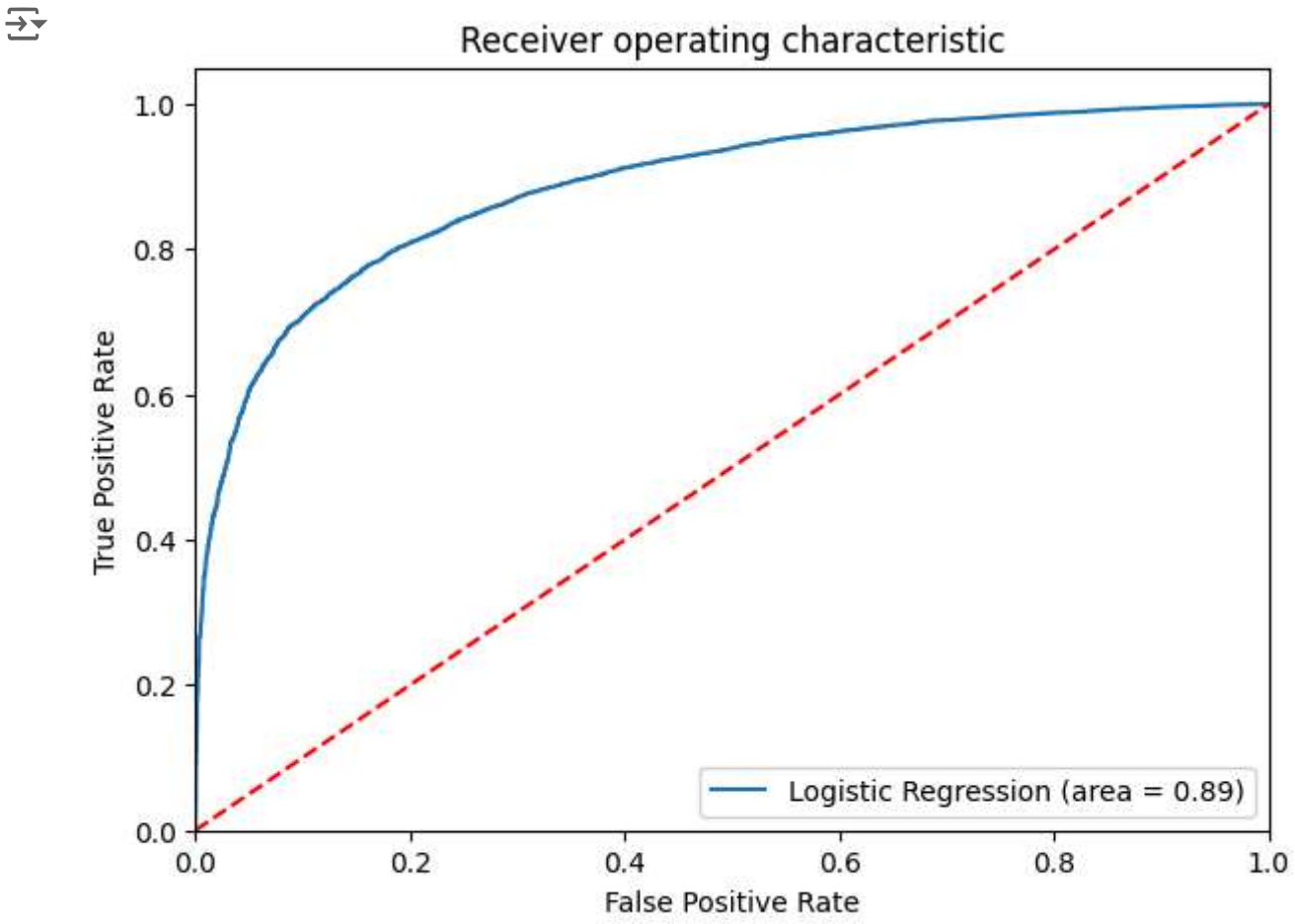
Model Performance Improvement

- Let's see if the f1_score can be improved further by changing the model threshold
- First, we will check the ROC curve, compute the area under the ROC curve (ROC-AUC), and then use it to find the optimal threshold
- Next, we will check the Precision-Recall curve to find the right balance between precision and recall as our metric of choice is f1_score

ROC Curve and ROC-AUC

- ROC-AUC on training set

```
logit_roc_auc_train = roc_auc_score(y_train, lg3.predict(X_train3))
fpr, tpr, thresholds = roc_curve(y_train, lg3.predict(X_train3))
plt.figure(figsize=(7, 5))
plt.plot(fpr, tpr, label="Logistic Regression (area = %0.2f)" % logit_roc_auc_train)
plt.plot([0, 1], [0, 1], "r--")
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Receiver operating characteristic")
plt.legend(loc="lower right")
plt.show()
```



- Logistic Regression model is giving a good performance on training set.

Optimal threshold using AUC-ROC curve

```
# Optimal threshold as per AUC-ROC curve
# The optimal cut off would be where tpr is high and fpr is low
fpr, tpr, thresholds = roc_curve(y_train, lg3.predict(X_train3))

optimal_idx = np.argmax(tpr - fpr)
optimal_threshold_auc_roc = thresholds[optimal_idx]
print(optimal_threshold_auc_roc)
```

```
0.7579399407330089
```

Checking model performance on training set

```
# checking model performance for this model
log_reg_model_train_perf_threshold_auc_roc = model_performance_classification_statsmodels(
    lg3, X_train3, y_train, threshold=optimal_threshold_auc_roc
)
print("Training performance:")
log_reg_model_train_perf_threshold_auc_roc
```

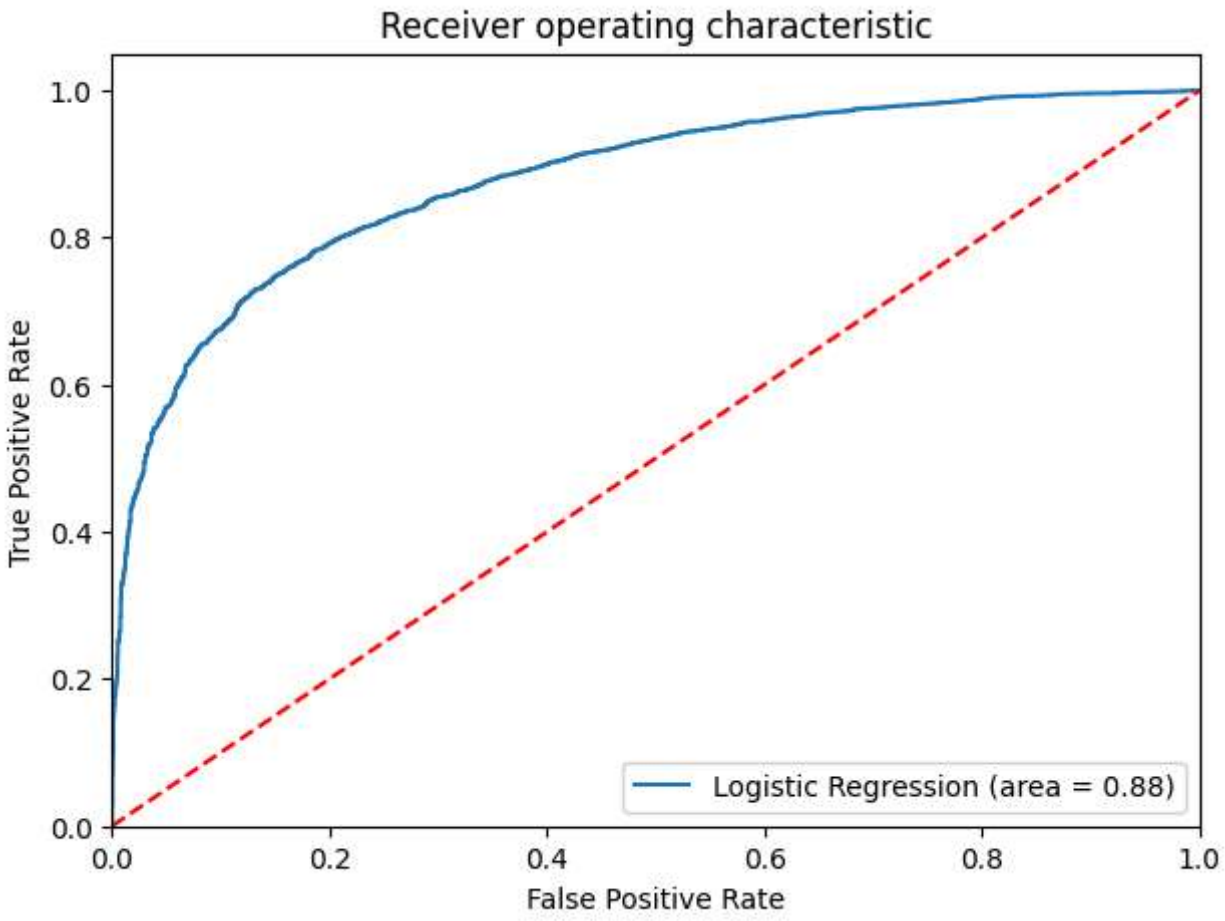
```
Training performance:
```

	Accuracy	Recall	Precision	F1
0	0.792499	0.777301	0.938383	0.85028

- Precision of model has increased but the other metrics have reduced.
- The model is still giving a good performance.

Checking model performance on test set

```
logit_roc_auc_train = roc_auc_score(y_test, lg3.predict(X_test3))
fpr, tpr, thresholds = roc_curve(y_test, lg3.predict(X_test3))
plt.figure(figsize=(7, 5))
plt.plot(fpr, tpr, label="Logistic Regression (area = %0.2f)" % logit_roc_auc_train)
plt.plot([0, 1], [0, 1], "r--")
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Receiver operating characteristic")
plt.legend(loc="lower right")
plt.show()
```




```
# checking model performance for this model
log_reg_model_test_perf_threshold_auc_roc = model_performance_classification_statsmodels(
    lg3, X_test3, y_test, threshold=optimal_threshold_auc_roc
)
print("Test performance:")
log_reg_model_test_perf_threshold_auc_roc
```

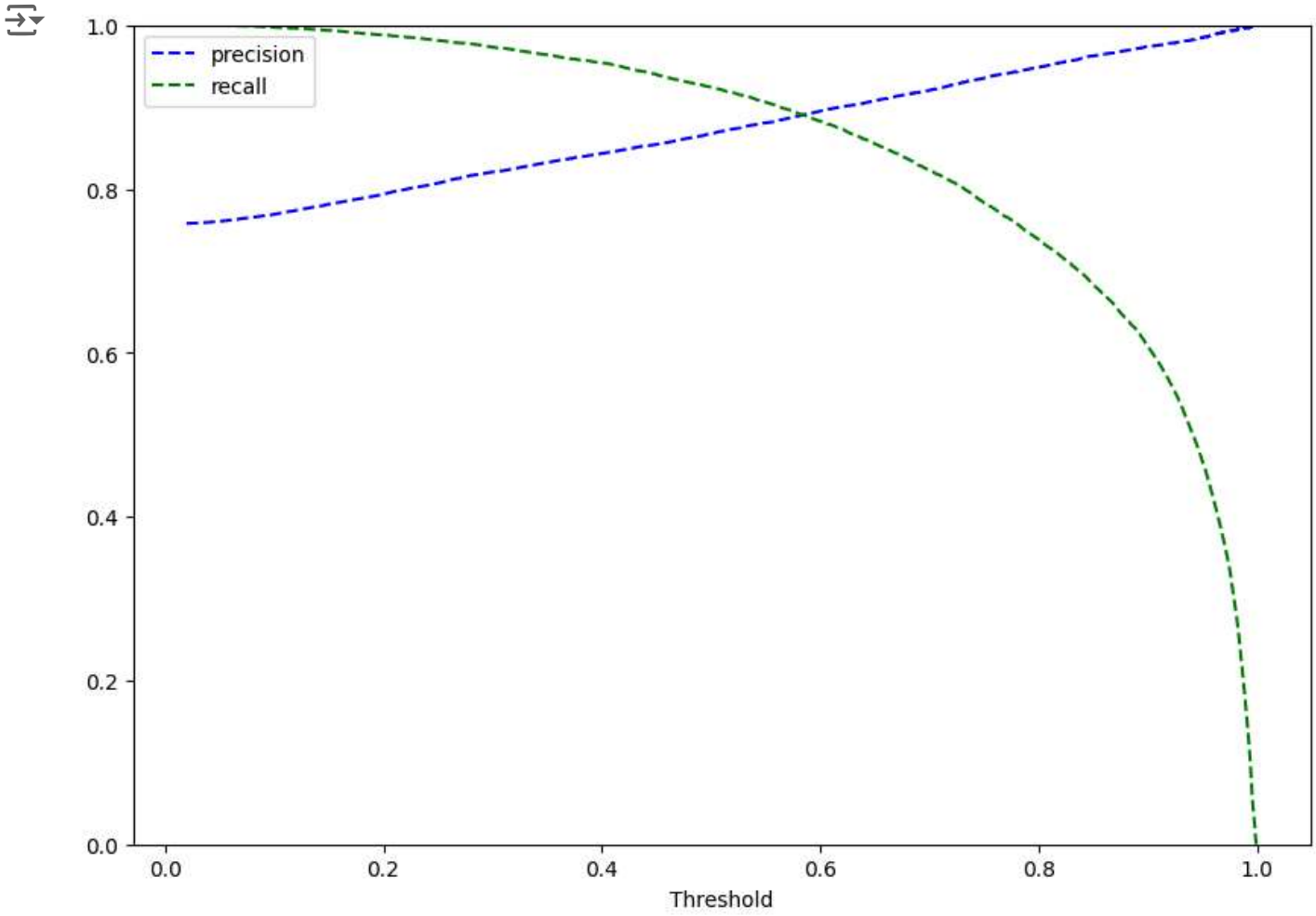
Test performance:

	Accuracy	Recall	Precision	F1
0	0.783299	0.771901	0.931774	0.844336

Precision-Recall Curve

```
y_scores = lg3.predict(X_train3)
prec, rec, tre = precision_recall_curve(y_train, y_scores,)


def plot_prec_recall_vs_tresh(precisions, recalls, thresholds):
    plt.plot(thresholds, precisions[:-1], "b--", label="precision")
    plt.plot(thresholds, recalls[:-1], "g--", label="recall")
    plt.xlabel("Threshold")
```

```
    plt.legend(loc="upper left")
    plt.ylim([0, 1])


plt.figure(figsize=(10, 7))
plot_prec_recall_vs_tresh(prec, rec, tre)
plt.show()
```



- At the threshold of 0.58, we get balanced recall and precision.

```
# setting the threshold
optimal_threshold_curve = 0.58
```

Checking model performance on training set

```
log_reg_model_train_perf_threshold_curve = model_performance_classification_statsmodels(
    lg3, X_train3, y_train, threshold=optimal_threshold_curve
)
print("Training performance:")
log_reg_model_train_perf_threshold_curve
```

```
Training performance:
```

	Accuracy	Recall	Precision	F1
0	0.833999	0.892301	0.889107	0.890701

- Model is performing well on training set.
- There's not much improvement in the model performance as the default threshold is 0.50 and here we get 0.58 as the optimal threshold.

Checking model performance on test set

```
log_reg_model_test_perf_threshold_curve = model_performance_classification_statsmodels(
    lg3, X_test3, y_test, threshold=optimal_threshold_curve
)
print("Test performance:")
log_reg_model_test_perf_threshold_curve
```

```
Test performance:
```

	Accuracy	Recall	Precision	F1
0	0.825512	0.884807	0.88588	0.885343

Model Performance Comparison and Final Model Selection

```
# training performance comparison

models_train_comp_df = pd.concat(
    [
        log_reg_model_train_perf.T,
        log_reg_model_train_perf_threshold_auc_roc.T,
        log_reg_model_train_perf_threshold_curve.T,
    ],
    axis=1,
)
models_train_comp_df.columns = [
    "Logistic Regression-default Threshold (0.5)",
    "Logistic Regression-0.76 Threshold",
    "Logistic Regression-0.58 Threshold",
]
```

```
print("Training performance comparison:")
models_train_comp_df
```

```
Training performance comparison:
```

	Logistic Regression-default Threshold (0.5)	Logistic Regression-0.76 Threshold	Logistic Regression-0.58 Threshold
Accuracy	0.835536	0.792499	0.833999
Recall	0.924048	0.777301	0.892301
Precision	0.867602	0.938383	0.889107
F1	0.894936	0.850280	0.890701

```
# testing performance comparison

models_test_comp_df = pd.concat(
    [
        log_reg_model_test_perf.T,
        log_reg_model_test_perf_threshold_auc_roc.T,
        log_reg_model_test_perf_threshold_curve.T,
    ],
    axis=1,
)
models_test_comp_df.columns = [
    "Logistic Regression-default Threshold (0.5)",
    "Logistic Regression-0.76 Threshold",
    "Logistic Regression-0.58 Threshold",
]

print("Test set performance comparison:")
models_test_comp_df
```

```
Test set performance comparison:
```

	Logistic Regression-default Threshold (0.5)	Logistic Regression-0.76 Threshold	Logistic Regression-0.58 Threshold
Accuracy	0.830635	0.783299	0.825512
Recall	0.917239	0.771901	0.884807
Precision	0.867838	0.931774	0.885880
F1	0.891855	0.844336	0.885343

- Almost all the three models are performing well on both training and test data without the problem of overfitting
- The model with a default threshold (0.5) is giving the best F1 score. Therefore it can be selected as the final model