**Assignment 1**

**Customer Subscription with Machine Learning**

**Name:- Nimisha Malviya**

**M.Sc.(Data Science) 2nd Year**

**Institute:- Banasthali Vidyapith,Rajasthan**

**Date:- 4 Oct 2023**

**Assignment 1: Customer Subscription Prediction with Machine Learning**

**Introduction:**

The banking industry plays a pivotal role in the economy by offering a wide range of financial services to individuals and businesses. One of the critical services provided by banks is the term deposit, a type of savings account with a fixed term and interest rate. Banks often engage in marketing campaigns to encourage customers to subscribe to term deposits. However, these campaigns can be costly and time-consuming, and their success depends on targeting the right customers.

This project aims to leverage data science and machine learning to optimise the marketing efforts of a Portuguese bank. By analysing a dataset containing information about previous marketing campaigns, customer attributes, and subscription outcomes, we seek to build a predictive model that can identify potential customers who are more likely to subscribe to a term deposit. Such a model can help the bank focus its marketing resources on the most promising leads, ultimately improving the efficiency of its campaigns and increasing subscription rates.

**Problem Statement:**

The problem at hand is twofold:

**Predictive Modelling:** Can we build an accurate predictive model that determines whether a customer will subscribe to a term deposit based on historical data and customer attributes? The primary goal is to develop a classification model that can distinguish between subscribers (positive class) and non-subscribers (negative class).

**Identifying Influential Factors:** What are the most important factors that influence a customer's decision to subscribe to a term deposit? Understanding the key drivers behind subscription decisions can provide valuable insights for the bank's marketing strategy and customer targeting efforts.

By addressing these challenges, this project aims to provide actionable recommendations to the bank, helping them allocate resources effectively, improve campaign outcomes, and enhance their overall marketing strategy in the context of term deposits.

## **About The Dataset**

**Given dataset:- The dataset you will be using is the Bank Marketing dataset:**

**https://archive.ics.uci.edu/ml/datasets/Bank+Marketing, which contains**

**information about customers of a Portuguese bank who were contacted by**

**telemarketers about a term deposit**

The dataset is related with direct marketing campaigns (phone calls) of a Portuguese banking institution.The classification goal of this dataset is to predict if the client or the customer of a polish banking institution will subscribe to a term deposit product of the bank or not.

**What is a Term Deposit?**

A term deposit is a cash investment held at a financial institution. Your money is invested for an agreed rate of interest over a fixed amount of time, or term. Term deposits can be invested into a bank, building society or credit union.

When the money is deposited, the customer understands that the money is there for the predetermined period which usually ranges from 1 month to 5 years and the interest rate is guaranteed not to change for that nominated period of time. Typically, the money can only be withdrawn at the end of the period – or earlier with a penalty attached.

Term deposits are popular with investors who prefer capital security and a set return as opposed to the fluctuations of, say, the share market. Many investors also use term deposits as a part of their investment mix.

**Attributes**

**Here is the description of all the variables :**

* Variable: Definition
* ID: Unique client ID
* age: Age of the client
* job: Type of job
* marital: Marital status of the client
* education: Education level
* default: Credit in default.
* housing: Housing loan
* loan: Personal loan
* contact: Type of communication
* month: Contact month
* day\_of\_week: Day of week of contact
* duration: Contact duration
* campaign: number of contacts performed during this campaign to the client
* pdays: number of days that passed by after the client was last contacted
* previous: number of contacts performed before this campaign
* poutcome: outcome of the previous marketing campaign

##### **Output variable (desired target):**

* Subscribed (target): has the client subscribed a term deposit?

## **Tools and Algorithms Used for Analysis**

* Python
* Numpy
* Pandas
* Scikit learn
* Seaborn
* matplotlib
* Logistic Regression
* KNN Classifier
* Decision Tree Classifier
* Random Forest Classifier
* Bagging
* Adaboost Classifier

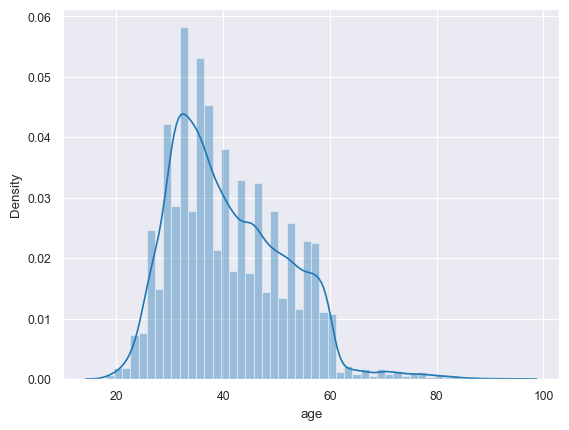
# **Q1. What is the distribution of the customer ages?**

In [446]:

sns.distplot(df['age'])

Out[446]:

<AxesSubplot:xlabel='age', ylabel='Density'>



In [447]:

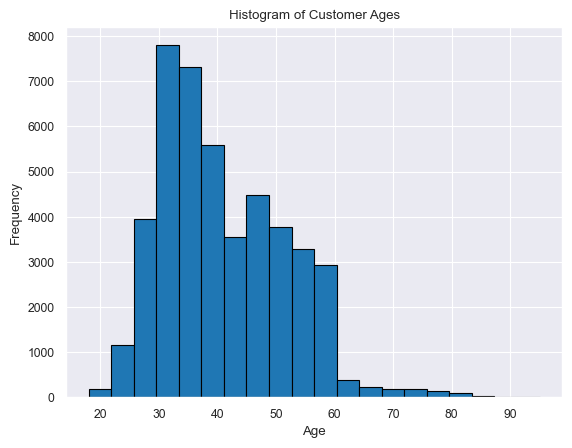
plt.hist(df['age'], bins**=**20, edgecolor**=**'k')

plt.xlabel('Age')

plt.ylabel('Frequency')

plt.title('Histogram of Customer Ages')

plt.show()



In [448]:

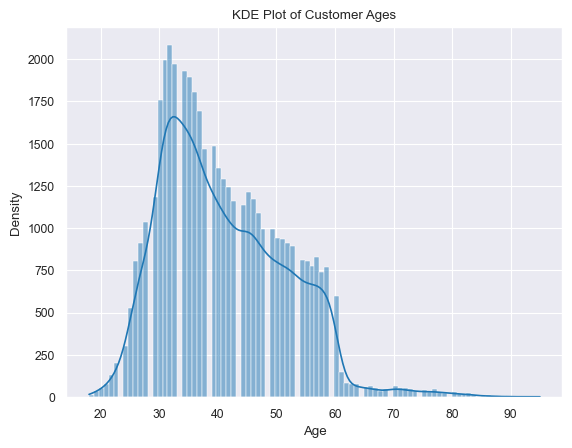
sns.histplot(df['age'], kde**=True**)

plt.xlabel('Age')

plt.ylabel('Density')

plt.title('KDE Plot of Customer Ages')

plt.show()



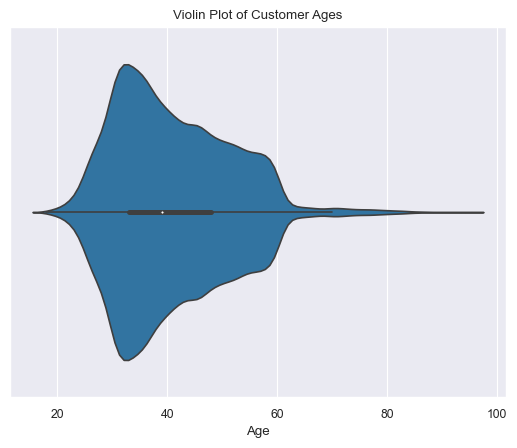
In [409]:

sns.violinplot(x**=**'age', data**=**df)

plt.xlabel('Age')

plt.title('Violin Plot of Customer Ages')

plt.show()

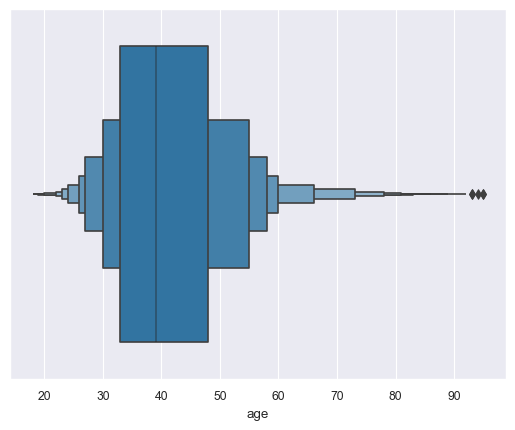


In [449]:

sns.boxenplot(df['age'])

Out[449]:

<AxesSubplot:xlabel='age'>



Each of these plots provides a different perspective on the distribution of customer ages, allowing you to choose the one that best suits your analysis and presentation needs.

# **Q2.) Relationship between customer age and subscription**

In [464]:

print(pd.crosstab(df['age'],df['y']))

y no yes

age

18 5 7

19 24 11

20 35 15

21 57 22

22 89 40

.. .. ...

90 0 2

92 0 2

93 0 2

94 1 0

95 1 1

[77 rows x 2 columns]

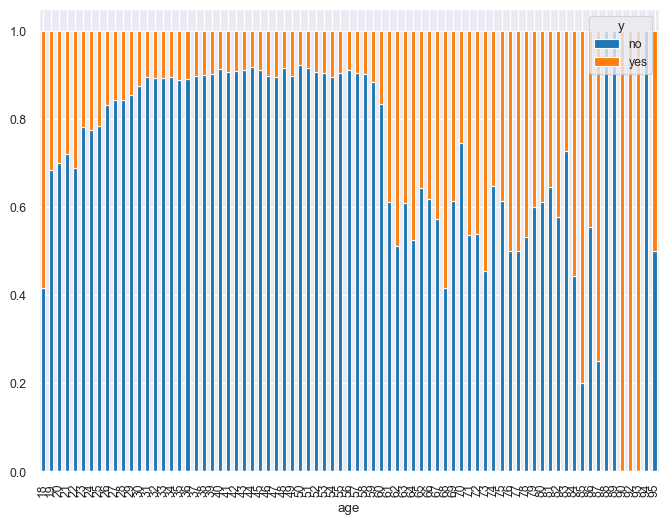
In [465]:

job **=** pd.crosstab(df['age'],df['y'])

job\_norm **=** job.div(job.sum(1).astype(float), axis**=**0)

In [466]:

job\_norm.plot.bar(stacked**=True**,figsize**=**(8,6));



# **Q3) Are there any other factors that are correlated with subscription?**

# to perform a more comprehensive analysis of factors correlated with subscription, you can:

# Calculate and visualise the correlation between numerical features and subscription status.

# Create cross-tabulations (contingency tables) for categorical features like job, marital status, etc., and

# analyse the distribution of subscriptions within each category. Use statistical tests like chi-squared tests for independence to determine if categorical features are significantly associated with subscription.

# **Correlation matrix**

In [478]:

corr **=** df.corr()

corr

Out[478]:

|  | **age** | **balance** | **day** | **duration** | **campaign** | **pdays** | **previous** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **age** | 1.000000 | 0.097783 | -0.009120 | -0.004648 | 0.004760 | -0.023758 | 0.001288 |
| **balance** | 0.097783 | 1.000000 | 0.004503 | 0.021560 | -0.014578 | 0.003435 | 0.016674 |
| **day** | -0.009120 | 0.004503 | 1.000000 | -0.030206 | 0.162490 | -0.093044 | -0.051710 |
| **duration** | -0.004648 | 0.021560 | -0.030206 | 1.000000 | -0.084570 | -0.001565 | 0.001203 |
| **campaign** | 0.004760 | -0.014578 | 0.162490 | -0.084570 | 1.000000 | -0.088628 | -0.032855 |
| **pdays** | -0.023758 | 0.003435 | -0.093044 | -0.001565 | -0.088628 | 1.000000 | 0.454820 |
| **previous** | 0.001288 | 0.016674 | -0.051710 | 0.001203 | -0.032855 | 0.454820 | 1.000000 |

In [479]:

fig,ax**=** plt.subplots()

fig.set\_size\_inches(20,10)

sns.heatmap(corr, annot**=True**, cmap**=**'viridis')

Out[479]:

<AxesSubplot:>



# **Create Cross-Tabulations for Categorical Features:**

In [480]:

*# Cross-tabulation for 'job' and subscription*

job\_crosstab **=** pd.crosstab(df['job'], df['y'])

​

*# Cross-tabulation for 'marital' and subscription*

marital\_crosstab **=** pd.crosstab(df['marital'], df['y'])

​

*# Print or visualize the cross-tabulations*

print(job\_crosstab)

print(marital\_crosstab)

y 0 1

job

admin. 4540 631

blue-collar 9024 708

entrepreneur 1364 123

housemaid 1131 109

management 8157 1301

retired 1748 516

self-employed 1392 187

services 3785 369

student 669 269

technician 6757 840

unemployed 1101 202

unknown 254 34

y 0 1

marital

divorced 4585 622

married 24459 2755

single 10878 1912

# **Use Chi-Squared Tests for Independence:**

In [481]:

**from** scipy.stats **import** chi2\_contingency

​

*# Perform chi-squared test for 'job' and subscription*

chi2, p, \_, \_ **=** chi2\_contingency(job\_crosstab)

​

*# Print the chi-squared statistic and p-value*

print(f'Chi-Squared Statistic: {chi2}')

print(f'P-Value: {p}')

Chi-Squared Statistic: 836.1054877471965

P-Value: 3.337121944935502e-172

# **Q4. What is the accuracy of the logistic regression model?**

Accuracy= Total Number of Predictions/Number of Correct Predictions

Accuracy is a measure of a model's overall correctness in making predictions.

It calculates the proportion of correctly classified instances (samples) out of the total instances in the dataset.

Accuracy is suitable for balanced datasets but can be misleading in imbalanced datasets.

​

The accuracy of the logistic Regression model is 89%

# **Q 5. What are the most important features for the logistic regression model?**

In [498]:

*# Get the coefficients of the logistic regression model*

coefficients **=** LR.coef\_[0]

​

*# Create a DataFrame to associate feature names with coefficients*

coefficients\_df **=** pd.DataFrame({'Feature': X.columns, 'Coefficient': coefficients})

​

*# Sort the DataFrame by absolute coefficient values in descending order*

coefficients\_df['Abs\_Coefficient'] **=** abs(coefficients\_df['Coefficient'])

coefficients\_df **=** coefficients\_df.sort\_values(by**=**'Abs\_Coefficient', ascending**=False**)

​

*# Visualise the most important features*

plt.figure(figsize**=**(10, 6))

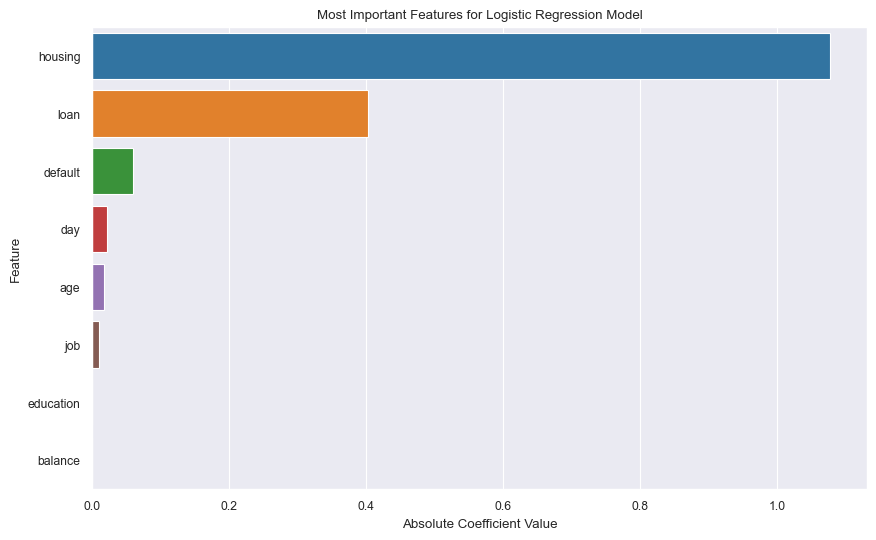
sns.barplot(x**=**'Abs\_Coefficient', y**=**'Feature', data**=**coefficients\_df)

plt.xlabel('Absolute Coefficient Value')

plt.ylabel('Feature')

plt.title('Most Important Features for Logistic Regression Model')

plt.show()



# From above graph we get that the most important features are housing,loan,default,day,age,job especially - housing , loan and default are the most significant

# **Q6. What is the precision of the logistic regression model?**

Precision:Precision measures the model's ability to correctly identify positive instances among the instances it predicts as positive (true positives) out of all predicted positive instances. It is especially relevant when there is a cost associated with false positives. Formula: Precision=True Positives/(True Positives + False Positives)

Precision for class 0 (negative class) is 1.00.

# **Q7. What is the recall of the logistic regression model?**

Recall (Sensitivity or True Positive Rate):Recall measures the model's ability to correctly identify all positive instances (true positives) out of all actual positive instances. It is especially relevant when missing positive cases is costly or critical. Formula: Recall=True Positives/(True Positives + False Negatives)

Recall for class 0 (negative class) is 0.89.

# **Q8. What is the f1-score of the logistic regression model?**

F1-Score:The F1-score is the harmonic mean of precision and recall, providing a balance between the two metrics. It is a useful metric when you want to consider both false positives and false negatives. Formula: F1-Score=2*Precision*Recall/(Precision + Recall)

F1-score is particularly useful when dealing with imbalanced datasets. F1-score for class 0 (negative class) is 0.94.

# **Q9. How can you improve the performance of the logistic regression model?**

# **To improve the performance of the logistic regression model, you can consider various techniques:**

1) Feature engineering: Select and engineer features that are more informative.

2)Hyperparameter tuning: Optimise hyperparameters like regularisation strength.

3)Handling class imbalance: If there's class imbalance in your data, consider techniques like oversampling or undersampling.

4)Trying different models: Experiment with other classification algorithms to see if they perform better. Here is the implementation-

In [499]:

**import** numpy **as** np

**import** pandas **as** pd

**from** sklearn.model\_selection **import** train\_test\_split, GridSearchCV, StratifiedKFold

**from** sklearn.linear\_model **import** LogisticRegression

**from** sklearn.preprocessing **import** StandardScaler

**from** imblearn.over\_sampling **import** RandomOverSampler

**from** sklearn.metrics **import** classification\_report, confusion\_matrix, accuracy\_score, f1\_score

​

*# Load your dataset (replace 'X' and 'y' with your features and target variable)*

*# X = df.drop('y', axis=1)*

*# y = df['y']*

​

*# Split the data into training and testing sets*

*#X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=1)*

​

*# Standardise numerical features (mean=0, std=1)*

scaler **=** StandardScaler()

X\_train\_scaled **=** scaler.fit\_transform(X\_train)

X\_test\_scaled **=** scaler.transform(X\_test)

​

*# Handle imbalanced data using Random Oversampling*

oversampler **=** RandomOverSampler(sampling\_strategy**=**'minority', random\_state**=**1)

X\_train\_resampled, y\_train\_resampled **=** oversampler.fit\_resample(X\_train\_scaled, y\_train)

​

*# Hyperparameter tuning using GridSearchCV*

param\_grid **=** {'C': [0.001, 0.01, 0.1, 1, 10], 'penalty': ['l1', 'l2']}

lr **=** LogisticRegression(max\_iter**=**1000, random\_state**=**1)

grid\_search **=** GridSearchCV(lr, param\_grid, cv**=**StratifiedKFold(n\_splits**=**5, shuffle**=True**, random\_state**=**1), scoring**=**'f1', n\_jobs**=-**1)

grid\_search.fit(X\_train\_resampled, y\_train\_resampled)

​

*# Get the best hyperparameters*

best\_lr **=** grid\_search.best\_estimator\_

​

*# Evaluate the model on the test set*

y\_pred **=** best\_lr.predict(X\_test\_scaled)

​

*# Print evaluation metrics*

print("Accuracy:", accuracy\_score(y\_test, y\_pred))

print("F1-Score:", f1\_score(y\_test, y\_pred))

print("Confusion Matrix:\n", confusion\_matrix(y\_test, y\_pred))

print("Classification Report:\n", classification\_report(y\_test, y\_pred))

Accuracy: 0.6184016514302566

F1-Score: 0.27036932619114745

Confusion Matrix:

[[7429 4584]

[ 592 959]]

Classification Report:

precision recall f1-score support

0 0.93 0.62 0.74 12013

1 0.17 0.62 0.27 1551

accuracy 0.62 13564

macro avg 0.55 0.62 0.51 13564

weighted avg 0.84 0.62 0.69 13564

# **Q10. What are the limitations of the logistic regression model?**

# **Limitations of the logistic regression model include:**

* Linearity assumption: Logistic regression assumes a linear relationship between the features and the log-odds of the response variable.
* Limited expressiveness: Logistic regression may not capture complex relationships in the data.
* Sensitive to outliers: Outliers can have a significant impact on logistic regression.
* Assumption of independence: Logistic regression assumes that features are independent of each other, which may not hold in all cases