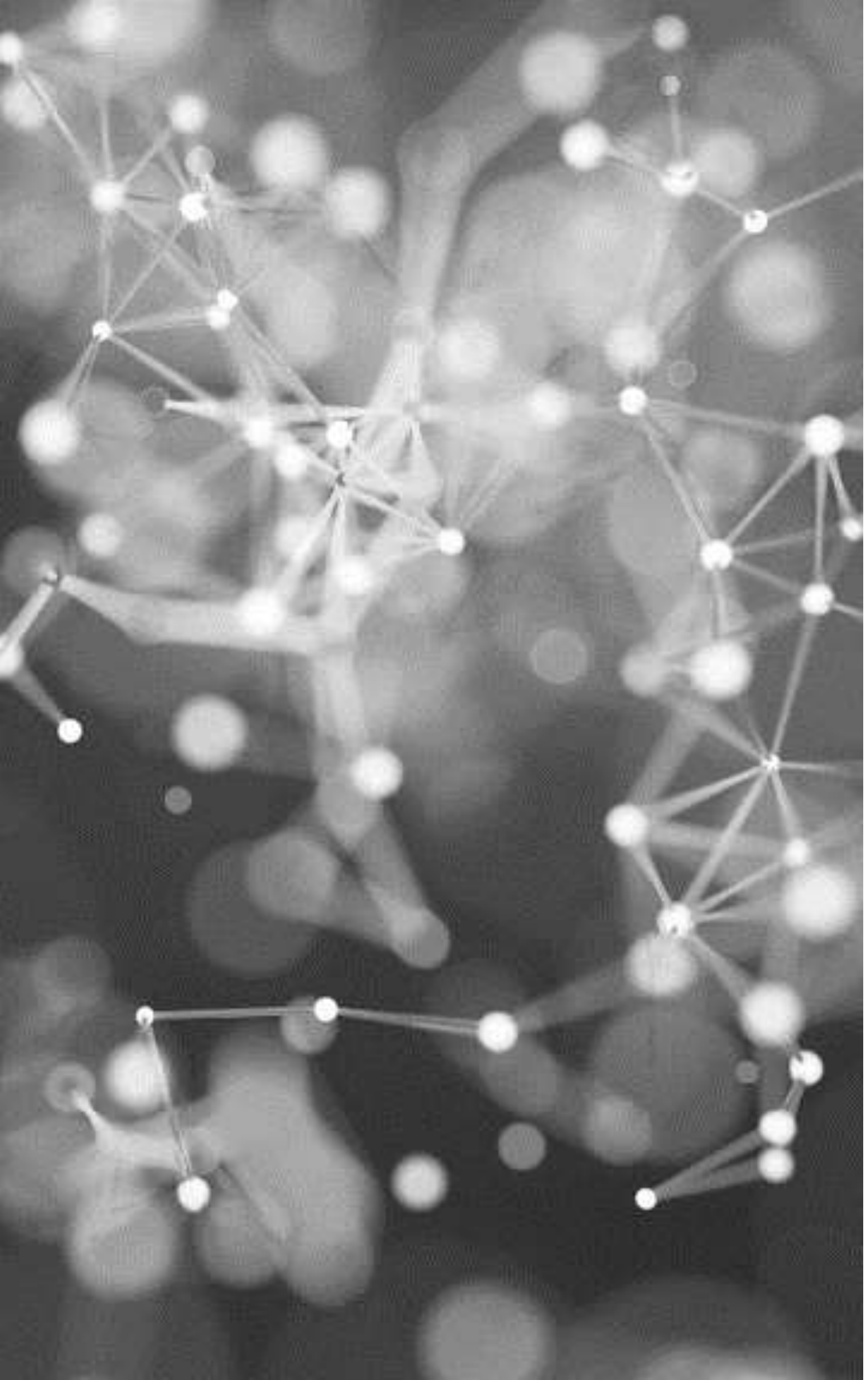


PORTUGUESE BANK MARKETING PREDICTION

ETL|MODEL BUILDING| PYTHON |POWER BI

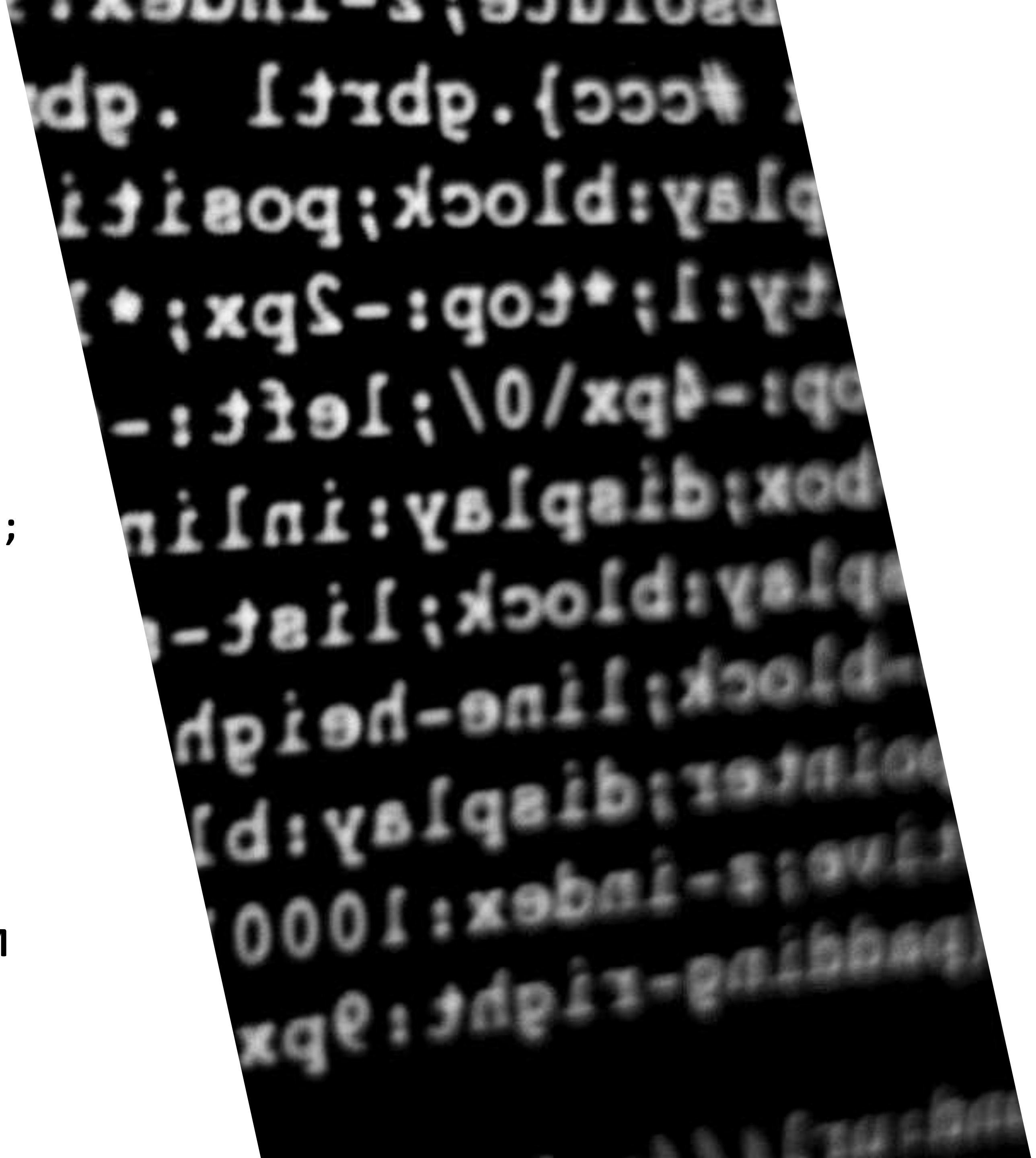
BUSINESS PROBLEM & MOTIVATION

- THE BANK WANTS TO INCREASE TERM DEPOSIT SUBSCRIPTIONS THROUGH TARGETED MARKETING.

- TELEPHONIC CAMPAIGNS ARE EXPENSIVE; CALLING UNINTERESTED CUSTOMERS REDUCES ROI.

- A PREDICTIVE MODEL CAN HELP PRIORITIZE POTENTIAL SUBSCRIBERS.

- GOAL: PREDICT WHETHER A CLIENT WILL SUBSCRIBE TO A TERM DEPOSIT BASED ON PAST CAMPAIGN DATA. BUSINESS PROBLEM & MOTIVATION



PAST WORK / LITERATURE

- TRADITIONAL MARKETING RELIED ON GENERIC CUSTOMER SEGMENTATION.
- PREVIOUS APPROACHES LACKED DATA-DRIVEN PERSONALIZATION.
- RECENT RESEARCH SUGGESTS USING ML FOR LEAD SCORING IMPROVES CAMPAIGN EFFICIENCY.
- PAST KAGGLE/ACADEMIC PROJECTS USED LOGISTIC REGRESSION, SVM, XGBOOST FOR SIMILAR TASKS.

ABOUT THE DATASET

- Dataset : Bank Marketing Data Set from UCI Repository.
- **Records:** 41,188 marketing campaign calls with 21 columns.
- **Target variable:** Whether the customer subscribed (y: yes/no).
- Includes client info, contact type, campaign outcome, economic context.

Why it's best:

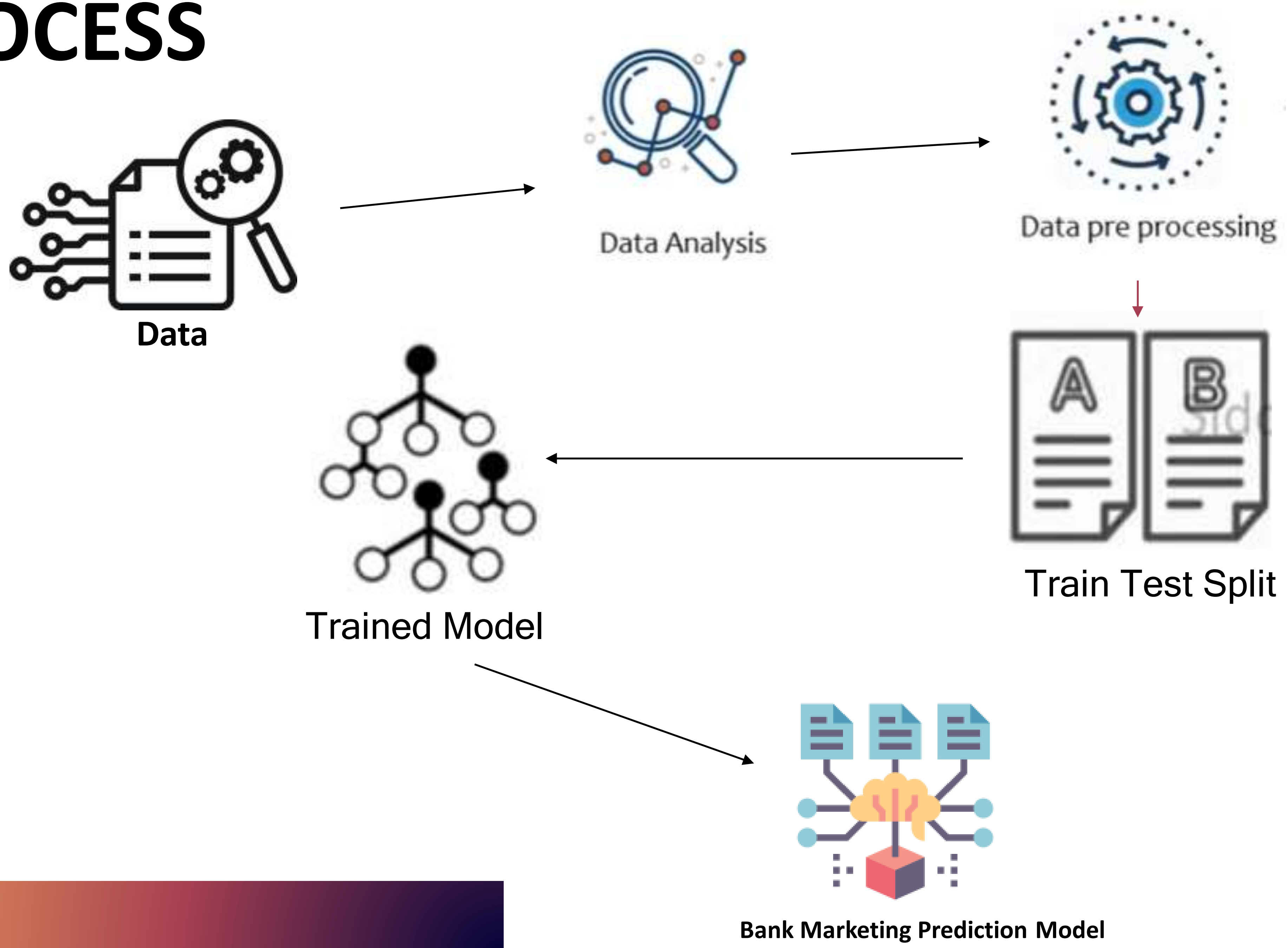
- It includes more predictive features **like pdays, previous, campaign, emp.var.rate, cons.price.idx, euribor3m, etc.,** which improve model performance.
- It uses cleaned and pre-processed data, according to the dataset documentation from UCI.
- Designed for better predictive modeling in mind, making it more suitable for machine learning tasks than the original dataset (bank-full.csv).

COLUMN DESCRIPTION

Column	Description
y (term deposit)	Whether the client subscribed to a term deposit (target variable).
age	Age of the client.
duration	Duration of the last contact (in seconds).
campaign	Number of contacts performed during this campaign.
pdays	Days since the client was last contacted (999 = never).
previous	Number of contacts performed before this campaign.
emp.var.rate	Employment variation rate (economic indicator).
cons.price.idx	Consumer price index (monthly).
cons.conf.idx	Consumer confidence index (monthly).
euribor3m	Euribor 3-month rate (daily).

nr.employed	Number of employees (quarterly).
contact_telephone	Indicates if contact was made via telephone.
job_*	One-hot encoded job type (e.g., job_student, job_admin.).
marital_*	One-hot encoded marital status (e.g., marital_single, marital_married).
education_*	One-hot encoded education level (e.g., education_university.degree).
default_yes	Indicates if the client has credit in default.
housing_yes	Indicates if the client has a housing loan.
loan_yes	Indicates if the client has a personal loan.
month_*	One-hot encoded last contact month (e.g., month_may, month_jul).
day_of_week_*	One-hot encoded weekday of last contact (e.g., day_of_week_mon).
poutcome_*	One-hot encoded outcome of previous campaign (e.g., poutcome_success).

PROCESS


Data
Data Analysis
Data pre processing
Train Test Split
Trained Model
Bank Marketing Prediction Model

ETL PROCESS &MODELING PROCESS

- Loaded raw CSV using Pandas.
- Handled 'unknown' values by removing affected rows.
- Encoded categorical columns using one-hot encoding.
- Saved the cleaned dataset to 'bank_cleaned.csv' for further processing.

- Target variable 'y' converted to binary (1: yes, 0: no).
- Balanced the data using SMOTE to handle class imbalance.
- Trained three models: Logistic Regression, Random Forest, XGBoost.
- Evaluated using Accuracy, Precision, Recall, and F1 Score.
- XGBoost performed best and was saved for predictions.

LOADING THE DATSET

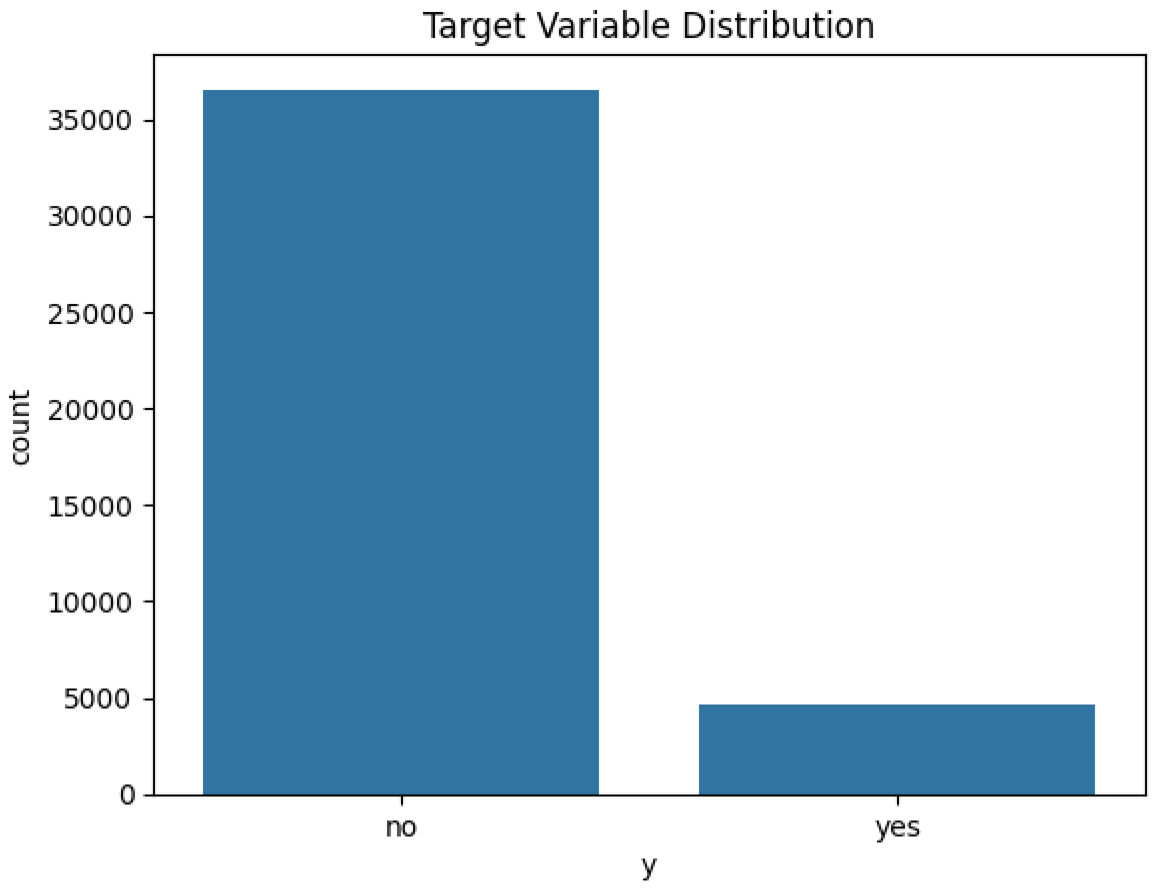
```
df = pd.read_csv('bank-additional-
full.csv', sep=';')
df.shape
df.head()
```


```
df = pd.read_csv('bank-additional-full.csv', sep=';')
df.shape
df.head()
```


	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previous	poutcome	emp.var.rate	cons.price.
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.
1	57	services	married	high.school	unknown	no	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.
2	37	services	married	high.school	no	yes	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.
3	40	admin.	married	basic.6y	no	no	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.
4	56	services	married	high.school	no	no	yes	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.

5 rows × 21 columns

INITIAL DATA EXPLORATION




```
# Target distribution
sns.countplot(x='y', data=df)
plt.title("Target Variable Distribution")
plt.show()

# Summary statistics
df.describe(include='all')
```

- This bar chart shows a significant class imbalance in the target variable y, with far more clients not subscribing to a term deposit than those who did.
- This imbalance may affect model performance and should be addressed using techniques like resampling or class weighting.

DATA CLEANING (ETL – TRANSFORM)

A. Handle 'unknown' values

```
for col in df.columns:
    if df[col].dtype == 'object':
        print(f"{col}: {df[col].value_counts().get('unknown', 0)} unknowns")
# Remove rows with any 'unknown' values
df = df[~df.isin(['unknown']).any(axis=1)]
```

B. Encode target variable

```
df['y'] = df['y'].map({'no': 0, 'yes': 1})
```

C. One-hot encode categorical columns

```
df_encoded = pd.get_dummies(df, drop_first=True)
```

D. Save Cleaned Data

```
cleaned_df = df_encoded.copy() cleaned_df.to_csv("bank_cleaned.csv", index=False)
from google.colab import files files.download("bank_cleaned.csv")
# Download cleaned CSV
```

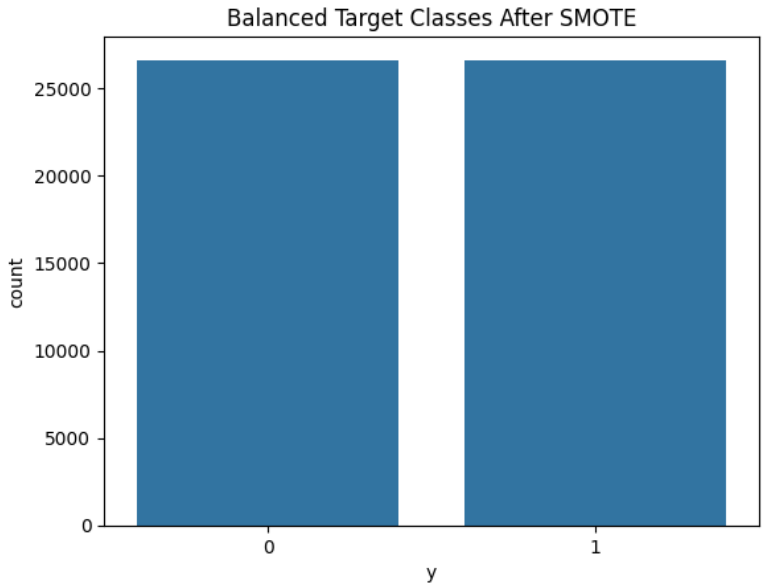
E. Feature-Target Split

```
X = cleaned_df.drop('y', axis=1)
y = cleaned_df['y']
```

F. Balance the Dataset (SMOTE)

```
sm = SMOTE(random_state=42)
X_resampled, y_resampled = sm.fit_resample(X, y)

# Check new class distribution
sns.countplot(x=y_resampled)
plt.title("Balanced Target Classes After SMOTE")
plt.show()
```



TRAINING & EVALUATION

A. Train/Test Split

```
X_train, X_test, y_train, y_test = train_test_split
(X_resampled, y_resampled, test_size=0.3, random_state=42)
```

B. Train Models

i. Logistic Regression

```
lr = LogisticRegression(max_iter=1000)
lr.fit(X_train, y_train)
```

ii. Random Forest

```
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
```

iii. XGBoost

```
xgb_model = xgb.XGBClassifier(use_label_encoder=False, eval_metric='logloss')
xgb_model.fit(X_train, y_train)
```

TRAINING & EVALUATION

C. Evaluate Models

```
# Create a function to compute all metrics
def get_metrics(model, X_test, y_test):
    y_pred = model.predict(X_test)
    return {
        "Accuracy": accuracy_score(y_test, y_pred),
        "Precision": precision_score(y_test, y_pred),
        "Recall": recall_score(y_test, y_pred),
        "F1 Score": f1_score(y_test, y_pred)
    }

# Compute metrics for all models
metrics = {
    "Logistic Regression": get_metrics(lr, X_test, y_test),
    "Random Forest": get_metrics(rf, X_test, y_test),
    "XGBoost": get_metrics(xgb_model, X_test, y_test)
}

# Convert to DataFrame
metrics_df = pd.DataFrame(metrics).T
metrics_df = metrics_df.round(3)  # Round for cleaner display

# Display the DataFrame
print("Model Performance Summary:")
display(metrics_df)
```

	Accuracy	Precision	Recall	F1 Score
Logistic Regression	0.926	0.933	0.918	0.925
Random Forest	0.950	0.943	0.958	0.950
XGBoost	0.946	0.946	0.946	0.946

- ***Among three classification models—Logistic Regression, Random Forest, and XGBoost—using Random Forest achieved the highest F1 Score (0.950), indicating a strong balance between precision and recall.***
- ***XGBoost also performed very well with balanced scores, while Logistic Regression had slightly lower but still strong performance***.

TRAINING & EVALUATION

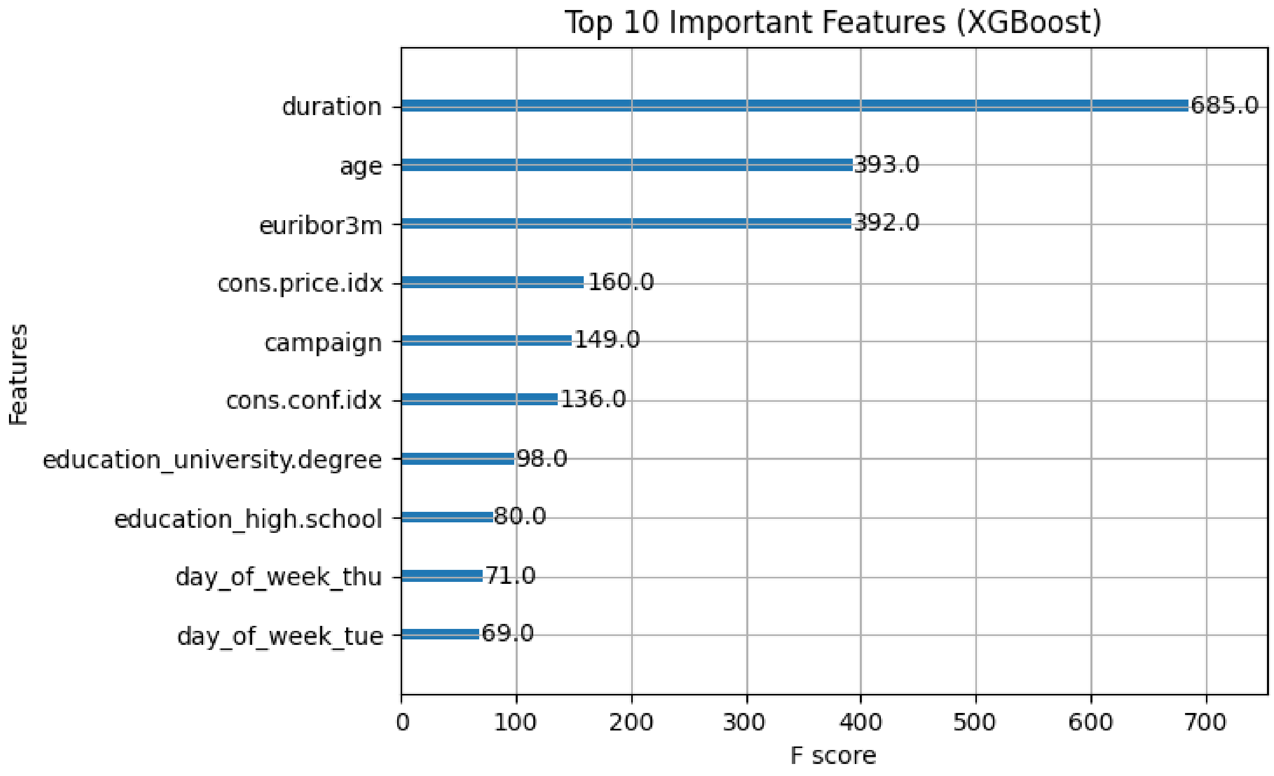
```
# Plot a bar chart for each metric
metrics_df.plot(kind='bar', figsize=(10, 6))
plt.title('Model Comparison: Logistic Regression vs Random Forest vs XGBoost')
plt.ylabel('Score')
plt.ylim(0, 1)
plt.xticks(rotation=0)
plt.legend(loc='lower right')
plt.grid(axis='y')
plt.show()
```

FEATURE IMPORTANCE – XGBOOST

```
xgb.plot_importance(xgb_model, max_num_features=10)
plt.title("Top 10 Important Features (XGBoost)")
plt.show()
```

Business Impact:

- This bar chart shows the top 10 most important features used by the XGBoost model to predict term deposit subscriptions.
- **duration** of the call is the most influential feature by far, followed by age and the euribor3m interest rate.
- Economic indicators like consumer price/confidence indexes and campaign-related variables also play a strong role.
- Education level and certain weekdays (Tuesday, Thursday) slightly contribute to the model's predictions as well.

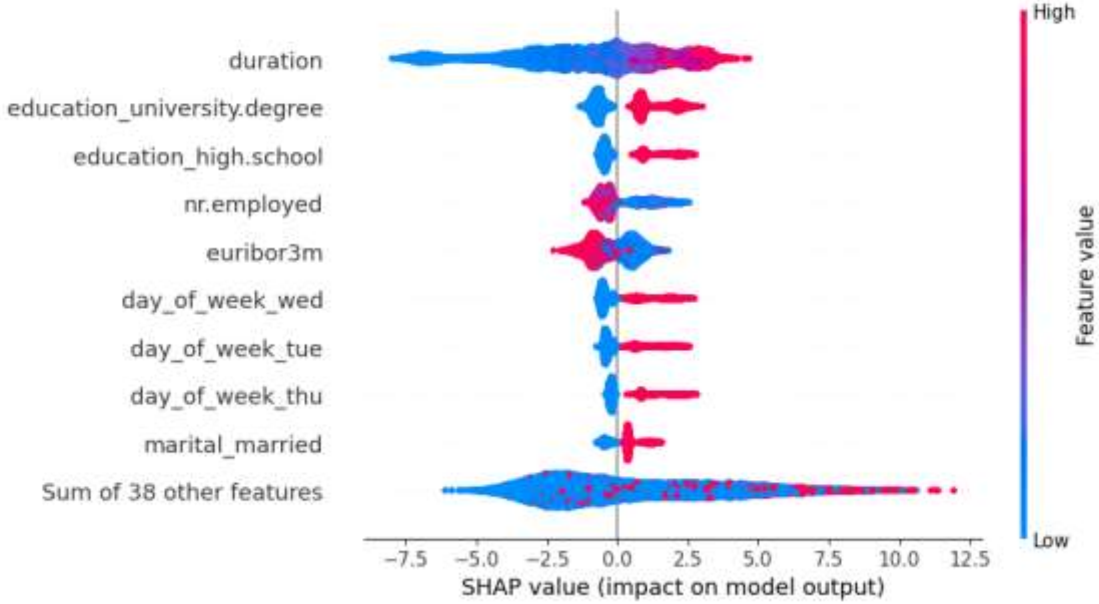


SHAP EXPLAINABILITY

```
explainer = shap.Explainer(xgb_model)
shap_values = explainer(X_test)
# SHAP Beeswarm Plot (Global feature impact)
shap.plots.beeswarm(shap_values)
```

Business Impact:

- SHapley Additive exPlanations (SHAP) summary plot explains how each feature influenced the XGBoost model's predictions for subscribing to a term deposit.
- duration has the strongest impact — longer calls (red) are associated with a higher likelihood of subscription.
- Features like education_university.degree, euribor3m, and nr.employed also influence predictions, with high/low values pushing the output positively or negatively.
- Each dot represents a client; the farther from zero, the more influence that feature had on the model's output.



FINAL DASHBOARD SUMMARY: BANK MARKETING CAMPAIGN OPTIMIZATION

1. KPI Cards
Total Customers – Count of all rows
Subscribed Customers – Count where actual = 1
High-Probability Leads – Count where subscription_probability ≥ 0.7
Filters: Used "Visual level filter" to isolate values like actual = 1

2. Pie Chart – Subscription Outcome
Visual Type: Pie Chart
Fields Used:
Legend: actual
Values: Count of actual
Purpose: Shows % of customers who subscribed vs. did not

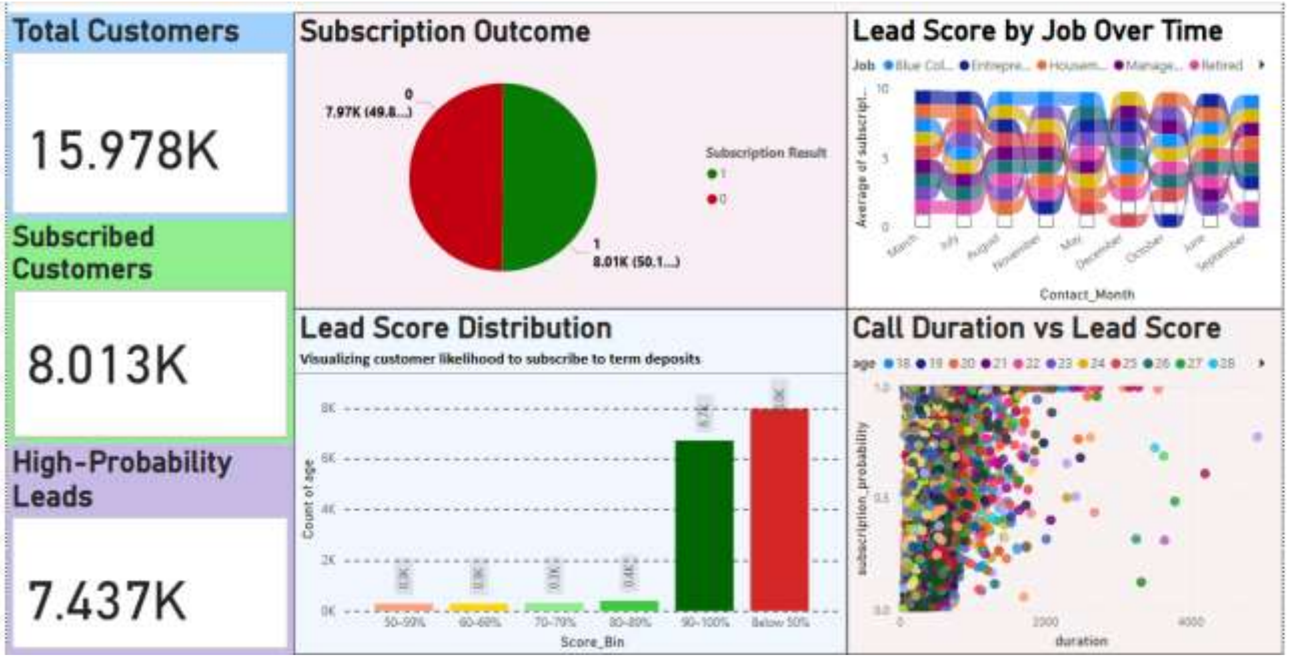
3. Bar Chart – Lead Score Distribution
Visual Type: Clustered Column Chart
X-axis: Score_Bin (you created using DAX to group probabilities)
Y-axis: Count of age (represents customer count)
Purpose: Visual ranking of how many customers fall in each prediction bucket

4. Ribbon Chart – Job Performance Over Time
Visual Type: Ribbon Chart
X-axis: Contact_Month
Legend: Job
Values: Average of subscription_probability
Filtered To: Only show subscription_probability ≥ 0.7
Purpose: Shows which job groups consistently rank highest in lead score across months

5. Scatter Plot – Call Duration vs Lead Score
Visual Type: Scatter Chart
Fields Used:
X-axis: duration
Y-axis: subscription_probability
Details: age
Tooltips: Education_Level, actual
Filtered: Optional filter on subscription_probability ≥ 0.5
Purpose: Explore whether longer calls result in higher lead scores

FINAL DASHBOARD SUMMARY: BANK MARKETING CAMPAIGN OPTIMIZATION

- Dataset Used: **bank_predictions.csv** (cleaned dataset with subscription_probability, actual, and categorical fields like Job, Education, Marital Status, etc.)
- Created Score_Bin, Job, Education_Level, Marital_Status, and Contact_Month using DAX.
- Explored subscription trends using Seaborn and Matplotlib.
- Visualized target balance before and after SMOTE.
- Created Power BI dashboard with KPI cards, pie chart, score distribution bar, customer table, filters, ribbon chart, and scatter plot.
- Dashboard helps identify high-score leads and optimize campaign strategy.



THANK YOU

Noor Ahamed Vempalle