LIVE PROJECT REPORT

Table of Contents

Predicting Profitability of Events Using Azure Machine Learning

1)Introduction

In today's dynamic business environment, accurately forecasting the profitability of contractual events is critical for minimizing financial risk and improving strategic decision-making. This project leverages **Azure Machine Learning** and **Power BI** to build a predictive model that determines whether a contract will result in a **net profit or loss**, based on historical financial data.

2) Business Problem Statement

This model aims to predict whether an event or contract will result in a **net profit or a net loss**. It assists the business in proactively identifying **financially risky ventures**. The approach involves analyzing the revenue generated from contracts and subtracting all associated costs, including:

- Agent or artist payouts
- Event-related expenses
- Commissions and overheads

  The primary goal is to **flag loss-making events before execution**, allowing for better **resource allocation**, **risk mitigation**, and **data-driven planning**.

3) Why Azure Machine Learning?

Azure Machine Learning is a robust cloud-based platform that streamlines the entire machine learning lifecycle. It was selected for this project due to:

- Easy data preprocessing and model training workflows
- Scalable compute environments
- AutoML and hyperparameter tuning capabilities

- Seamless integration with Python notebooks and Power BI
- Model versioning, tracking, and deployment pipelines

Azure ML empowers both data scientists and business users to **collaborate efficiently** and deploy models into production reliably.

4) Methodology and Implementation Steps

i) Data Preparation

```
import pandas as pd
from sqlalchemy import create_engine, text
import urllib

# DB credentials
server = 'qaececrm-s1.database.windows.net'
database = 'QAECECRM_June2025'
username = 'dbadmin'
password = 'DashTech1234'
driver = 'ODBC Driver 17 for SQL Server'

# Connection string
params = urllib.parse.quote_plus(
    f"DRIVER={driver};SERVER={server};DATABASE={database};UID={username};PWD={password}"
)
engine = create_engine(f"mssql+pyodbc:///?odbc_connect={params}")
```

- ✓ To connect Azure Machine Learning Studio with our SQL Server database, we used a secure ODBC connection string leveraging SQLAlchemy and the pyodbc driver. This allowed us to extract contract-level financial data directly from the Azure-hosted database (QAECECRM_June2025) for preprocessing and model training within the Azure ML environment.
- ✓ To construct the training dataset, a comprehensive SQL query was executed to aggregate relevant financial and contract-level information. This query joined multiple tables—including agent payroll logs, transaction costs, contract deposits, artist rosters, and contract terms—to calculate key metrics such as total credits received, total payouts, deposits received, and ultimately, the **net profit or loss per contract**. A categorical label (Profit, Loss, or Break-Even) was also derived to serve as the target variable for model training. This pre-aggregated dataset enabled a holistic view of each contract's financial outcome, forming the foundation for predictive modeling in Azure ML

To enhance the model's predictive capabilities, several **derived features** were engineered from the raw financial data:

```python
import pandas as pd
import numpy as np

# Create Profit Margin (handle divide by zero)
JoinedData['ProfitMargin'] = JoinedData.apply(
    lambda row: row['NetProfitLoss'] / row['GrossContractAmount']
            if row['GrossContractAmount'] != 0 else 0, axis=1)

# Binary Loss Indicator
JoinedData['IsLoss'] = JoinedData['ProfitLossStatus'].apply(lambda x: 1 if x == 'Loss' else 0)

# High-value contract flag (example threshold: 1 million)
JoinedData['IsHighValueContract'] = JoinedData['GrossContractAmount'] > 1_000_000

# Credit to cost ratio (avoid divide by zero)
JoinedData['CreditToCostRatio'] = JoinedData.apply(
    lambda row: row['CreditsReceived'] / (row['TotalPaidOut'] + 1), axis=1)

# Agent's average profit and number of contracts
agent_stats = JoinedData.groupby('Agent')['NetProfitLoss'].agg(['mean', 'count']).reset_index()
agent_stats.columns = ['Agent', 'AgentAvgProfit', 'AgentContractCount']

# Merge back to main dataframe
JoinedData = JoinedData.merge(agent_stats, on='Agent', how='left')
```

- **Profit Margin**: A ratio of net profit/loss to the gross contract amount, with handling for divide-by-zero cases.
- **IsLoss**: A binary indicator flagging whether a contract resulted in a financial loss (1 = Loss, 0 = Profit or Break-Even).
- **IsHighValueContract**: A flag indicating whether the gross contract value exceeded $1 million, helping differentiate large-scale events.
- **Credit to Cost Ratio**: The ratio of credits received to total payouts, providing insight into the balance between income and expenses.
- **Agent-Level Metrics**: For each agent, we calculated their **average profit** and **number of contracts handled**, which were then merged back into the main dataset. These features help capture historical performance trends and agent effectiveness.

These engineered features added contextual depth to the dataset and improved the model's ability to generalize profitability trends across contracts.

ntractAmount	Agent	List_of_Artist	Venue_Name	Venue_city	LOB	Terms	CreditsReceived	TotalPaidOut	DepositsReceived	NetProfitLoss	ProfitLossStatus	ProfitMargin	IsLoss	IsHighValueContract	CreditToCostRatio	AgentAvgProfit
2300.0	471.0	BLACK & BLUE	ELK RIVER COUNTRY CLUB	banner elk	3.0	3	0.0	460.0	800.0	340.0	Profit	0.147826	0	False	0.0	806.703824
17000.0	435.0	PERFECT 10 BAND	BELMOND CHARLESTON PLACE	charleston	3.0	1	0.0	3400.0	8500.0	5100.0	Profit	0.300000	0	False	0.0	3399.283202
5500.0	330.0	THE MEN OF DISTINCTION	1208 WASHINGTON PLACE	columbia	3.0	3	0.0	1100.0	2750.0	1650.0	Profit	0.300000	0	False	0.0	1248.279375
500.0	125.0	BURKE	ADVENTURES UNLIMITED (800-662-0667)	ocoee	3.0	1	0.0	100.0	100.0	0.0	Break-Even	0.000000	0	False	0.0	1005.517904
3000.0	NaN	RISSE	CAMBERLEY BROWN HOTEL	louisville	3.0	1	0.0	0.0	1500.0	1500.0	Profit	0.500000	0	False	0.0	NaN

The dataset contains around (325632 rows and 18 columns)

```python
JoinedData['Agent'] = JoinedData['Agent'].fillna(-1)  # or 0 if you prefer
JoinedData['AgentAvgProfit'] = JoinedData['AgentAvgProfit'].fillna(0)
JoinedData['AgentContractCount'] = JoinedData['AgentContractCount'].fillna(0)
```

To ensure model stability and avoid issues related to missing values, we applied **missing data imputation**:

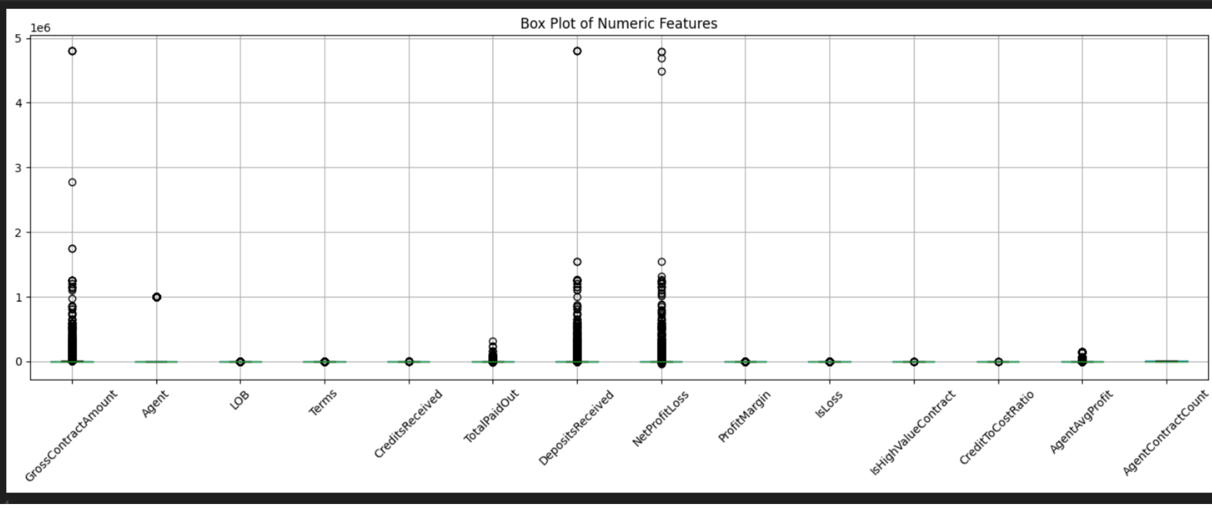
- **Missing Agent IDs** were replaced with -1 to represent unknown or unassigned agents.
- **Agent Average Profit** and **Agent Contract Count** were filled with 0, assuming that agents with no prior history should have neutral impact on the model.

This preprocessing step ensured that the machine learning algorithms could handle all rows uniformly without being affected by null values or incomplete agent data.

ii) Exploratory Data Analysis

```
import matplotlib.pyplot as plt

JoinedData.boxplot(figsize=(15, 6))
plt.title("Box Plot of Numeric Features")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



A box plot was created to visualize the distribution of numerical features in the dataset. This visualization helps identify the presence of outliers, skewness, and data spread across features.

Key observations:

- **GrossContractAmount**, **TotalPaidOut**, **DepositsReceived**, and **NetProfitLoss** show significant outliers, which is expected due to a wide range in contract sizes and revenue streams.
- Features such as **ProfitMargin**, **CreditToCostRatio**, and **AgentAvgProfit** are relatively tightly distributed for most entries but still contain a few extreme values.
- Binary flags like **IsLoss** and **IsHighValueContract** appear as flat distributions, as expected.

```python
lower = JoinedData['GrossContractAmount'].quantile(0.05)
upper = JoinedData['GrossContractAmount'].quantile(0.95)

filtered_df = JoinedData[(JoinedData['GrossContractAmount'] >= lower) & (JoinedData['GrossContractAmount'] <= upper)]
```

```python
agent_lower = filtered_df['Agent'].quantile(0.05)
agent_upper = filtered_df['Agent'].quantile(0.95)

filtered_df = filtered_df[(filtered_df['Agent'] >= agent_lower) & (filtered_df['Agent'] <= agent_upper)]
```

To reduce the impact of extreme outliers, we applied quantile-based filtering on key numeric columns:

- For **GrossContractAmount**, we retained only the contracts that fall between the 5th and 95th percentile values. This helped remove unusually small or large contract amounts that could skew model performance.
- Similarly, we filtered the **Agent** feature to include only values within the 5th to 95th percentile range, ensuring that outlier agent IDs or encoding artifacts did not introduce noise.

iii) Model Building

In Addition to predicting contract level losses, a secondary objective was to predict the average probability of each agent using historical data. This will help to identify which agents typically contribute to profitable deals and can inform agent selection or contract negotiations.

```python
features = [
    'GrossContractAmount', 'CreditsReceived', 'TotalPaidOut', 'DepositsReceived',
    'ProfitMargin', 'IsHighValueContract', 'CreditToCostRatio',
    'AgentAvgProfit', 'AgentContractCount'
]
X = JoinedData[features].fillna(0)
y = JoinedData['IsLoss']
```

- ✓ Selected relevant features like revenue, costs, profit margin, and agent performance indicators.
- ✓ Defined the **binary target variable IsLoss**, indicating whether a contract resulted in a loss (1) or not (0)

However this part appears to be a placeholder and wasn't used directly in the regression model.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor

# Aggregate historical average profit per contract for each agent
agent_profit = filtered_df.groupby('Agent').agg({
    'NetProfitLoss': 'mean',
    'GrossContractAmount': 'mean',
    'CreditsReceived': 'mean',
    'TotalPaidOut': 'mean',
    'DepositsReceived': 'mean',
    'AgentContractCount': 'first'  # count is constant per agent
}).reset_index()

agent_profit.rename(columns={'NetProfitLoss': 'AvgProfit'}, inplace=True)

# Fill missing agents (-1) or NaNs if any
agent_profit = agent_profit.fillna(0)
```

- ✓ Aggregated historical contract data per agent to create new dataset.
- ✓ Computed metrics such as:
    - ➢ Average net profit (AvgProfit)
    - ➢ Average values for revenue and costs
    - ➢ Number of contracts handled (AgentContractCount)
- ✓ Filled missing or placeholder agents (-1) with neutral values (e.g., 0).

```
features = ['GrossContractAmount', 'CreditsReceived', 'TotalPaidOut', 'DepositsReceived', 'AgentContractCount']
target = 'AvgProfit'

X = agent_profit[features]
y = agent_profit[target]
```

- ✓ We chose the features that believed to influence on agent's profitability
- ✓ Target variable is the agent profit across their contracts.

iv) Model Evaluation

```
from sklearn.metrics import root_mean_squared_error, mean_absolute_error, r2_score
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)


rmse = root_mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

# Adjusted R²
n = X_test.shape[0]
p = X_test.shape[1]
adjusted_r2 = 1 - (1 - r2) * ((n - 1) / (n - p - 1))


# Print results
print("Test RMSE:", rmse)
print("Test MAE:", mae)
print("Test R² Score:", r2)
print("Adjusted R² Score:", adjusted_r2)
```

```
Test RMSE: 181.80113751626965
Test MAE: 113.3966043637459
Test R² Score: 0.9124194417877189
Adjusted R² Score: 0.8933801900024404
```

To assess how well the model predicts an agent's average profit based on contract features, the dataset was split into training and testing sets (80/20 split). A Random Forest Regressor was then trained and evaluated using key regression metrics.

Metric	Value	Interpretation
RMSE	181.80	On average, predictions deviate by ±$181.8 from the true agent profit values.
MAE	113.40	The average absolute prediction error is $113.4.
R^2 Score	0.912	91.2% of the variability in agent profitability is explained by the model.

Metric	Value	Interpretation
Adjusted R^2	0.893	Accounts for the number of predictors; strong model generalization.

Key Findings

```
# Group by Agent and summarize predictions
agent_summary = filtered_df.groupby('Agent').agg(
    TotalPredictedProfit=('PredictedProfit', 'sum'),
    AveragePredictedProfit=('PredictedProfit', 'mean'),
    NumberOfContracts=('PredictedProfit', 'count')
).reset_index()

# Sort to find top agents
top_agents = agent_summary.sort_values(by='TotalPredictedProfit', ascending=False).head(10)

print(top_agents)
[31]
```

```
     Agent  TotalPredictedProfit  AveragePredictedProfit  NumberOfContracts
0     -1.0          5.208112e+07              440.022983             118360
129  430.0          1.183200e+07             1312.333476               9016
121  417.0          1.173114e+07             1004.550250              11678
11   125.0          1.113537e+07              943.995008              11796
84   330.0          8.975029e+06             1242.390471               7224
57   253.0          8.935180e+06             1603.010383               5574
77   313.0          8.165792e+06             1473.703684               5541
74   307.0          7.061024e+06              885.950371               7970
39   203.0          5.844285e+06             1315.095530               4444
61   270.0          5.503124e+06              944.093940               5829
```

This summary allowed us to rank agents based on their predicted contribution to the company's bottom line. The top 10 agents, sorted by total predicted profit, were identified to highlight high performers who consistently contribute to profitable engagements.

These insights can inform strategic staffing, agent incentives, or resource allocation decisions.

```python
# Group by Agent and Venue to calculate total predicted profit
agent_venue_profit = top_agents_data.groupby(['Agent', 'Venue_Name'])['PredictedProfit'].sum().reset_index()

# Sort by profit within each agent and get top venue per agent
top_venue_per_agent = agent_venue_profit.sort_values(['Agent', 'PredictedProfit'], ascending=[True, False])
top_venue_per_agent = top_venue_per_agent.groupby('Agent').first().reset_index()
```

```python
print(top_venue_per_agent)
```

```
   Agent                             Venue_Name  PredictedProfit
0   -1.0                                    TBA    673780.469569
1  125.0                           CAROLINA INN    320931.658167
2  203.0  GROVE PARK INN & COUNTRY CLUB - GREAT HALL    398337.609385
3  253.0                  FARMINGTON COUNTRY CLUB    167396.936650
4  270.0                            CAROLINA CC    143743.296224
5  307.0                   CHARLOTTE COUNTRY CLUB    270098.191419
6  313.0                              [Unknown]    413670.711870
7  330.0                          POINSETT CLUB    299137.441638
8  417.0                  SIGMA PHI EPSILON HOUSE    103133.012440
9  430.0                     CAROLINA YACHT CLUB    314222.468814
```

These insights shows about the top venue with the predicted profit.

5) Deployment Pipeline

The trained model is saved using joblib.

```
pip install joblib
```

```
Requirement already satisfied: joblib in /anaconda/envs/jupyter_env/lib/python3.10/site-packages (1.5.1)
Note: you may need to restart the kernel to use updated packages.
```

```python
import joblib
joblib.dump(model, "output/model.pkl")
```

Scoring Script

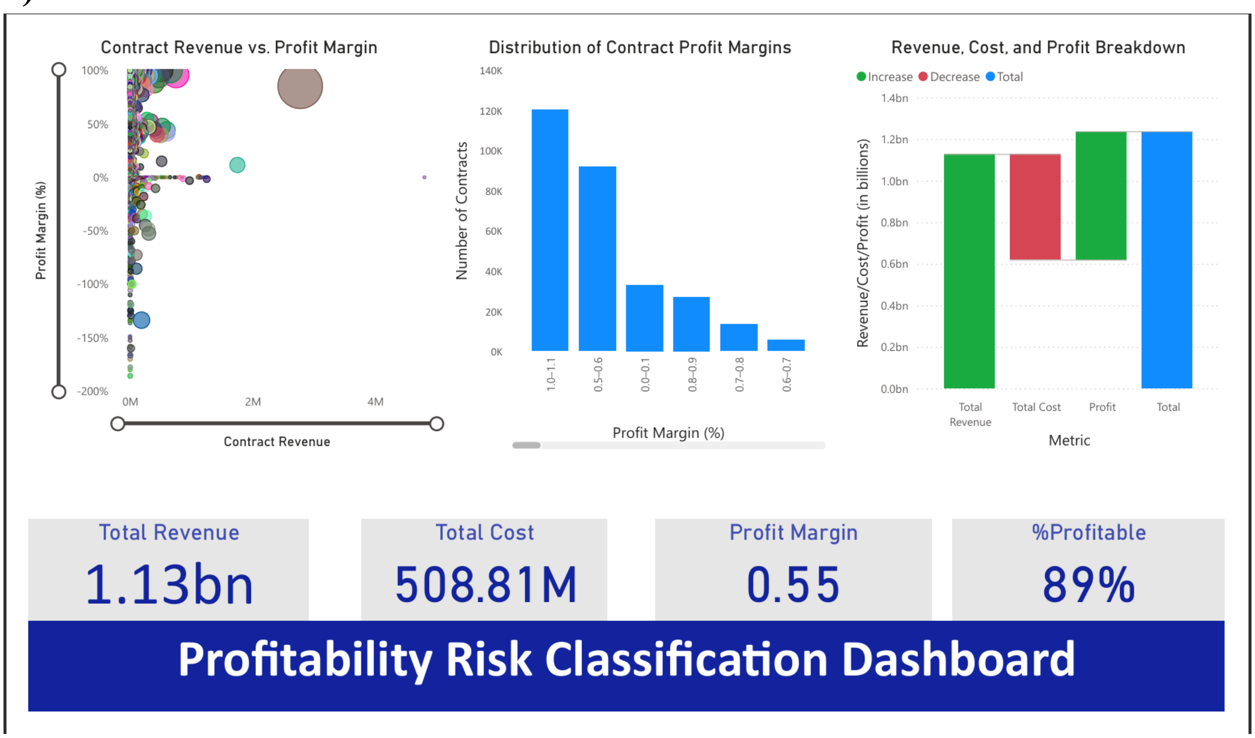
```
import json
import os
import joblib
import numpy as np
import pandas as pd

# Called once when the container starts
def init():
    global model
    model_path = os.path.join(
        os.getenv("AZUREML_MODEL_DIR"),    # /var/azureml-app/azureml-models/profit-regressor/1
        "model.pkl"
    )
    model = joblib.load(model_path)

# Called for every invocation
def run(raw_data):
    data = pd.read_json(raw_data)
    preds = model.predict(data)
    return json.dumps(preds.tolist())
```

This code is the **scoring script** used for model deployment in Azure ML. The init() function loads the trained model (model.pkl) from the Azure ML environment when the container starts. The run() function is triggered on each prediction request—it reads the incoming JSON data, uses the loaded model to make predictions, and returns the results as a JSON-formatted list. This script enables the model to serve real-time predictions via a REST API.

6) PowerBI Dashboard



The final model outputs and profitability insights were visualized using **Power BI Desktop**.

Key dashboard highlights include:

- **Total Revenue**: 1.13 billion
- **Total Cost**: 508.81 million
- **Profit Margin**: 55%
- **% Profitable Contracts**: 89%

Visual Elements:

- Contract Revenue vs. Profit Margin Scatter Plot
- Distribution of Profit Margins across all contracts
- Breakdown of Revenue, Cost, and Profit
- Profitability Risk Classification highlighting high-risk contracts

Power BI enables stakeholders to **interact with the results**, **filter by contract**, and **focus on underperforming events**, enhancing decision-making.

7) Conclusion

This project demonstrates how cloud-based machine learning and interactive visualization can transform contract-level financial analysis. By predicting profitability in advance:

- Businesses can **avoid unprofitable events**.
- **Strategic planning and budgeting improve**.
- **Human oversight becomes more data-driven**.

The integration of Azure ML and Power BI provides a **scalable and transparent solution** for real-time financial risk assessment.