# Machine Learning Deployment Project Report

**ANTL202 Final Project Deployment**

Author:  Casper Formadi and Ragaganbir Singh
Date: December 14, 2025

## 1. Executive Summary

This report outlines  our deployment architecture for the ANTL202 Final Project. The objective was to create a unified interface to serve multiple machine learning models using a web-based dashboard. The system utilizes Python and Streamlit to provide an interactive frontend, allowing users to input data and receive real-time loan predictions from four distinct pre-trained models.

## 2. System Architecture

The application is built as a single-page Streamlit application (app.py) acting as an inference engine. It loads pre-trained models into memory upon startup using caching mechanisms to ensure performance.

### Model Inventory

The system integrates four specific models, each serving a unique analytical purpose:

| Model File | Format | Function |
|---|---|---|
| regression_model.pkl | Pickle | Numerical Regression Analysis |
| best_loan_classifier_pipeline.joblib.gz | Joblib (GZip) | Loan Approval Prediction |
| classification_model.pkl | Pickle | General Text/Data Classification |
| deep_learning_model.h5 | Keras/HDF5 | Deep Learning Inference |

# 3. Deployment Challenges & Solutions

## The Large File Size Limit

A big challenge was encountered during the version control process. The primary loan classification model, originally named best_loan_classifier_pipeline.joblib, exceeded the GitHub web upload limit of 25MB so we needed to find workarounds like uploading the file as a .gz file

## Implementation of Compression Strategy

To resolve this without relying on Git Large File Storage (LFS) or external hosting which relies too much on permission control, a compression strategy was implemented:

1. **Compression:** The model was compressed to a .gz using 7-zip, reducing the file size significantly while preserving all serialized data. The file was renamed to best_loan_classifier_pipeline.joblib.gz then uploaded to github.
2. **Native Loading:** The application logic was updated to utilize joblib's native ability to handle compressed files. This eliminated the need for temporary decompression steps or additional code complexity.

# 4. Application Logic

The following logic was implemented in app.py to ensure robust error handling and automatic decompression:

```
# Logic for loading the compressed pipeline
try:
    # Joblib automatically detects .gz extension and decompresses
    models['loan_pipeline'] = joblib.load(
        'best_loan_classifier_pipeline.joblib.gz'
    )
except FileNotFoundError:
    # Fallback error handling
    models['loan_pipeline'] = None
    st.error("Model file not found.")
```

We also had to implement the RemainderColList manually as it was dropped from newer versions of scikit-learn using the app.py code itself.

# 5. Conclusion

The deployment pipeline is now fully functional. The Streamlit dashboard successfully interfaces with all four models, including the compressed loan classifier. The solution adheres to GitHub's storage constraints while maintaining fast inference times and a user-friendly experience.