PROJECT REPORT

LendWise

Smart Lender- Applicant Credibility Prediction for Loan Approval

Team Members:

- Saumya Goel 20BCE1574
- Aryan Guddetti 20BCE1590
- Karthik G 20BCE1021
- Josiah James 20BCE1344

1 INTRODUCTION

1.1 Overview

LendWise is an innovative smart lending system designed to revolutionize the process of loan applicants. Using predictive analytics powered by machine learning, LendWise enables lenders to accurately predict loan statuses during the loan application process. It automates the evaluation of borrower information, enabling lenders to make informed decisions with speed and precision.Harnessing the power of its extensive training, Lendwise swiftly analyzes the provided information and employs its deep understanding of loan credibility to assess the user's eligibility for a loan. The system takes into account various factors, including credit score, income stability, employment history, and existing debts, among others. By considering these crucial indicators, Lendwise generates a personalized prediction that accurately reflects the user's likelihood of loan approval.

1.2 Purpose

Lendwise serves a clear purpose: to simplify the loan application process and empower users with valuable insights into their loan eligibility. The primary objective of Lendwise is to enhance the loan eligibility assessment process. By providing users with accurate and reliable predictions, the website enables them to make informed financial decisions. With Lendwise, individuals can save time and effort by quickly obtaining insights into their loan prospects. The automated evaluation process eliminates the need for lengthy manual assessments, allowing users to receive instant feedback on their loan eligibility.
Furthermore, it aims to improve access to financial services by serving as a platform that connects borrowers with suitable loan providers based on their eligibility predictions.With its intelligent and user-centric approach, Lendwise aims to enhance the loan application experience and empower individuals to make informed financial choices.

2 LITERATURE SURVEY

2.1 Existing problem

The problem that Lendwise seeks to allay as a smart lender system iis to address the challenges of the loan application process by streamlining eligibility assessment and providing accurate predictions. This includes simplifying complex eligibility criteria, reducing the time and effort required for loan applications, and ensuring consistent and fair decision-making while offering personalized feedback to borrowers.

Now keeping this in mind,there are existing approaches and methods that are still commonly employed to solve the problem of loan eligibility prediction in a smart lender system like Lendwise.

- Rule-based systems still prevail that rely on a set of predefined rules to assess loan eligibility. These rules are typically based on industry standards, regulatory requirements, and specific lending policies. Loan applications are evaluated against these rules, and decisions are made based on whether the applicant meets the predetermined criteria.

- Evaluating the borrower's credit history is a common approach used to assess loan eligibility.Based on the credit history analysis, lenders make decisions regarding loan approval or rejection.
- Over recent times there have also been the rise of statistical models such as logistic regression, decision trees, and random forests that have been commonly used for loan eligibility prediction. These models analyze historical loan data, considering factors such as credit score, income, employment history, and debt-to-income ratio to predict the likelihood of loan approval.
- Machine learning algorithms, including support vector machines (SVM), neural networks, and gradient boosting, have shown promise in loan eligibility prediction. These algorithms can effectively capture complex patterns and relationships in large datasets, improving the accuracy of loan eligibility predictions.

And that is exactly what our LendWise model sought to research into whilst coming up with a more refined solution to this problem. It could try to incorporate credit scoring models to evaluate loan eligibility and risk while analyzing the loan history and characteristics of similar borrowers to identify patterns and make predictions about new loan application in a much more efficient manner. A combination of different techniques or customized approaches tailored to the unique characteristics of the lending domain may yield the best results.

2.2 Proposed solution

Our proposed solution for the loan eligibility prediction system involves a comprehensive evaluation of different machine learning techniques to identify the most accurate model with an accuracy value exceeding 95%. This will ensure highly reliable and precise loan eligibility predictions.

To implement this solution, we will leverage modern web frameworks, with a focus on using React for the frontend development. React frameworks like Formik, Yup, and modern CSS frameworks will be employed to enhance the user interface and provide seamless form handling, validation, and styling capabilities.

These are the key steps entailing our solution:

1. Data Collection and Preparation: A relevant dataset will be used that would include borrower information, credit history, income, employment details, and other pertinent factors. The dataset will undergo preprocessing steps such as data cleaning, feature selection, and transformation to ensure high-quality input for the machine learning models.

2. Model Training and Comparison: Multiple machine learning techniques will be employed, such as logistic regression, decision trees, random forests, KNN, and neural networks. Each model will be trained and evaluated using suitable performance metrics. The goal is to identify the model that achieves the highest accuracy, surpassing the target threshold of 95%.

3. Model Deployment and Integration: Once the most accurate model is identified, it will be deployed and integrated into the lending platform. The model will be incorporated into the backend system, which will handle loan eligibility predictions based on user inputs.

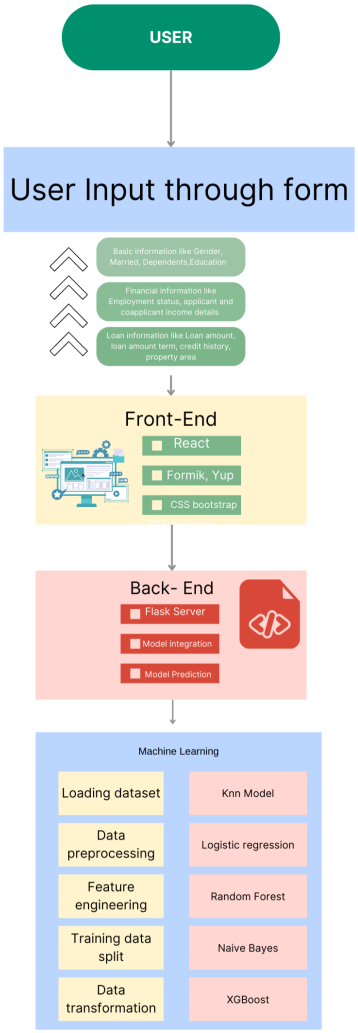
4. Frontend Development: The frontend of the application will be developed using React and relevant frameworks such as Formik for form handling and validation, Yup for schema-based form validation, and modern CSS frameworks to create an intuitive and visually appealing user interface. This will enhance the user experience and provide a seamless loan application process.

5. Testing and Validation: Rigorous testing and validation procedures will be performed to ensure the accuracy and reliability of the loan eligibility prediction system. Various test cases and real-world scenarios will be considered to assess the performance and robustness of the solution.

By implementing this proposed solution, we aim to deliver a highly accurate loan eligibility prediction system by comparing and selecting the best-performing machine learning model. The integration of modern frontend frameworks will enhance the user experience, making the loan application process smooth and intuitive.

3 THEORETICAL ANALYSIS

3.1 Block Diagram



3.2 Hardware / Software designing

Hardware Requirements:

- Computer or server with sufficient processing power and memory to handle the application workload.
- Adequate storage capacity to store the dataset, trained models, and other related files.
- Stable internet connection for data retrieval and model training, if applicable.

Software Requirements:

- Operating System (e.g., Windows,macOS) compatible with the chosen development tools.
- Integrated Development Environment (IDE) such as Visual Studio Code
- Programming language:
    - Python for backend development with Flask.
    - React.js framework for frontend development.

- Web development libraries and frameworks such as Formik and Yup for form handling and validation in React.
- Node.js and npm (Node Package Manager) for managing frontend dependencies.
- Flask for building the server-side logic and APIs.
- Modern CSS frameworks like Bootstrap or Tailwind CSS for styling and layout in the frontend.

4 EXPERIMENTAL INVESTIGATIONS

While working on this project, several analyses and investigations were conducted to ensure its effectiveness and accuracy. Here are some key areas of analysis:

- Feature Engineering: Various features were examined to determine their relevance and importance in predicting loan eligibility. Feature engineering techniques such as normalization, encoding categorical variables, and creating new derived features were applied to enhance the predictive power of the model.

- Model Selection and Evaluation: Multiple machine learning algorithms, including but not limited to logistic regression, decision trees, random forests, and XGBoost, were evaluated to identify the best-performing model.

Evaluation metrics such as accuracy were used to assess and compare the models' performance:

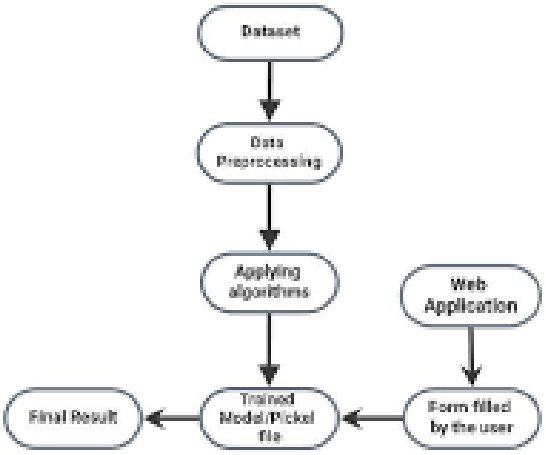
- XG Boost: Achieved an accuracy of 96.08% without grid search and 94.11% with grid search.
- Logistic Regression: Achieved an accuracy of 90.20%.
- Decision Tree: Achieved an accuracy of 84.313%.
- Naive Bayes: Achieved an accuracy of 84.313%.
- KNN (K-Nearest Neighbors): Achieved an accuracy of 78.4313%, but with grid search, the accuracy improved to 86.274%.
- Random Forest: Initially achieved an accuracy of 52.9411%, but with grid search, the accuracy improved to 94.11%.

These experimental findings indicate that XG Boost outperformed other models with the highest accuracy of 96.08% without grid search and 94.11% with grid search. Logistic Regression also performed well with an accuracy of 90.20%. However, Decision Tree, Naive Bayes, and KNN models achieved relatively lower accuracies, which improved for KNN after applying grid search. Random Forest initially had a low accuracy but showed a significant improvement after grid search.

These findings suggest that XG Boost is the most accurate model for the Lendwise smart lender system, providing reliable loan eligibility predictions. Further optimization and fine-tuning of the

models through grid search or other hyperparameter tuning techniques may improve the performance of the models and enhance their accuracy.

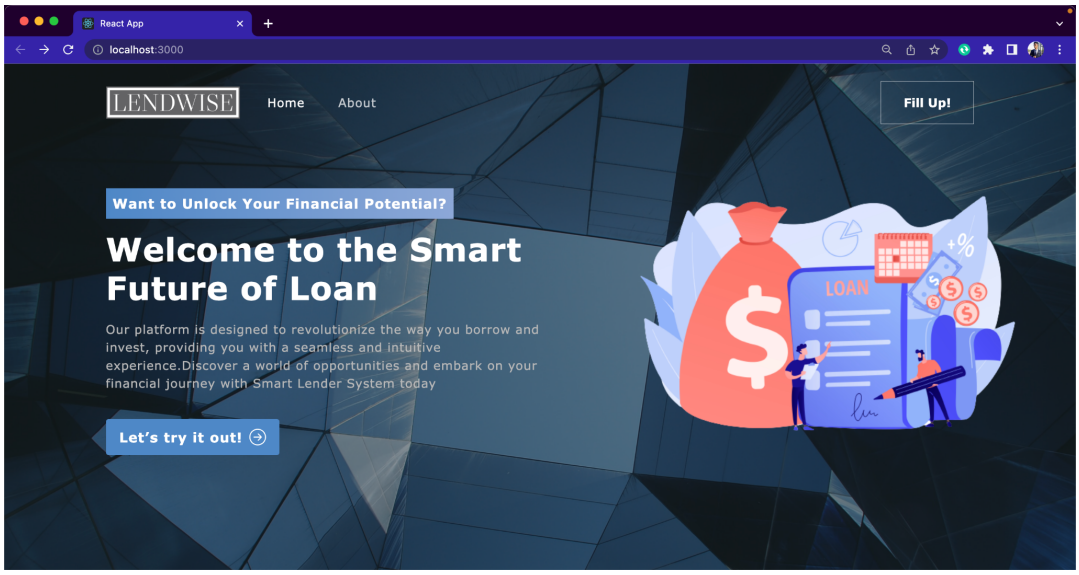
5 FLOWCHART



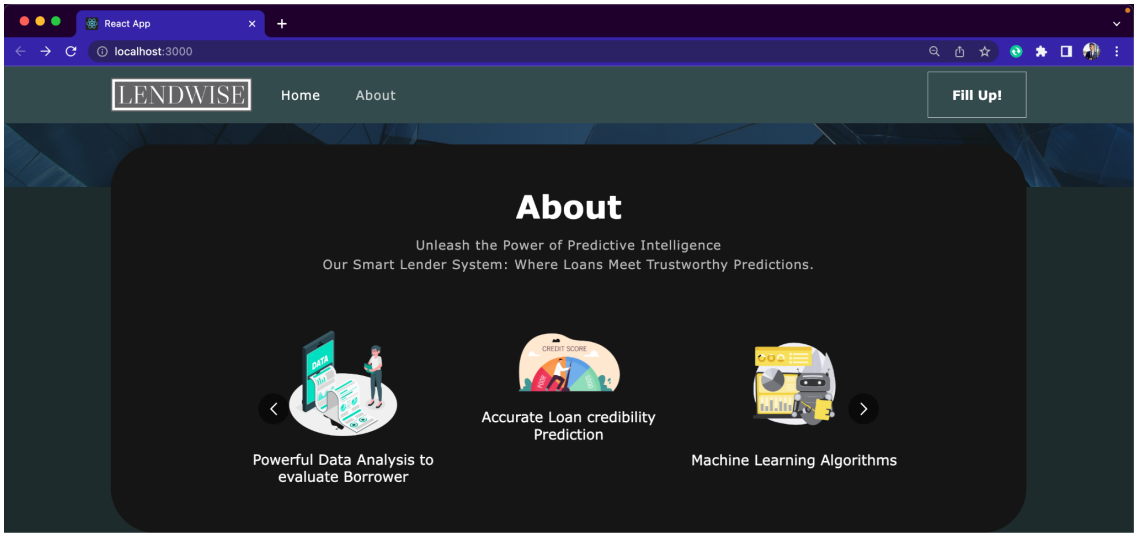
6 RESULT

Screenshots of our application:

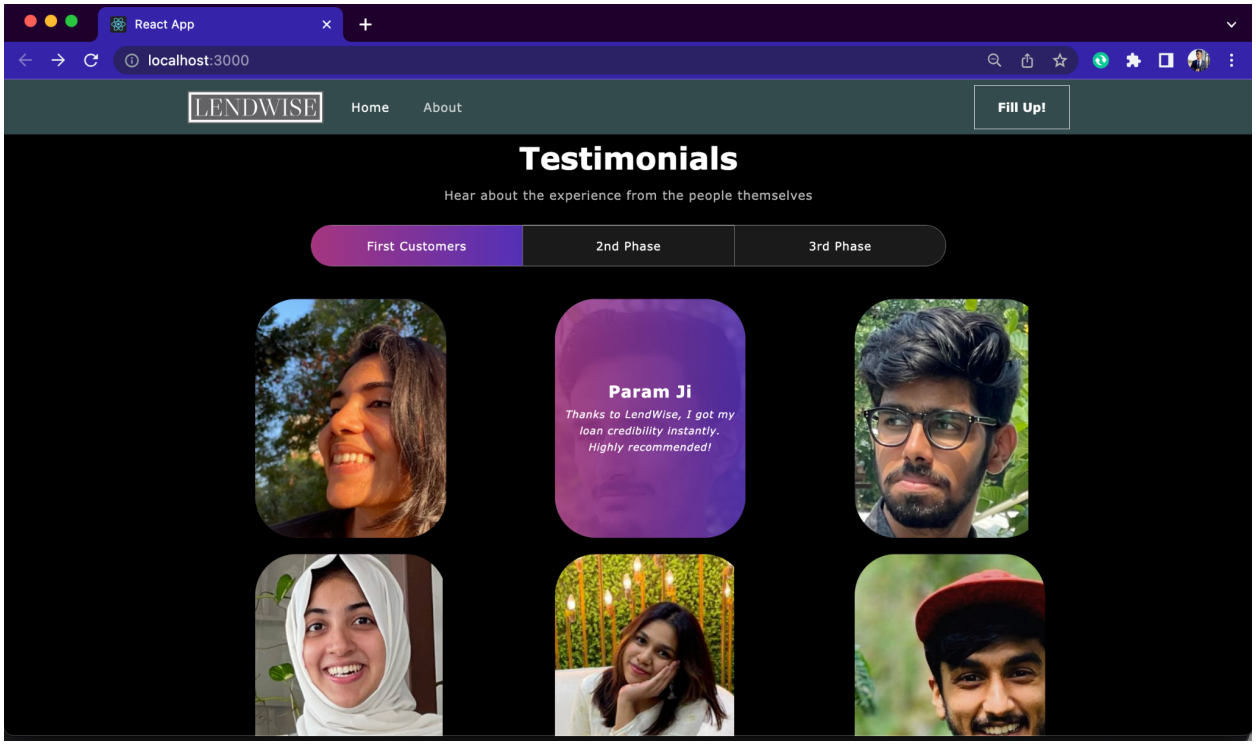
- Home Page
    - With custom brand logo for ‘LendWise”
    - Attractive banner front
    - Animated typing for the Main title that would change the last word between - loan, lending, and leverage



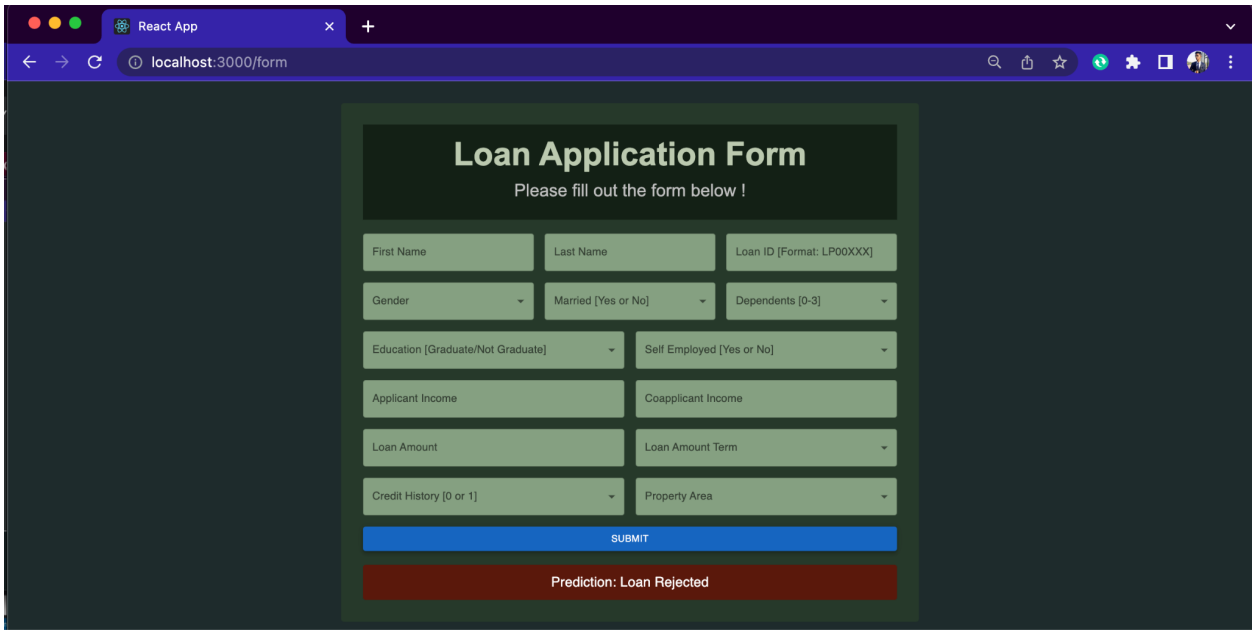
- About Section
    - Slider window with moving icons and text

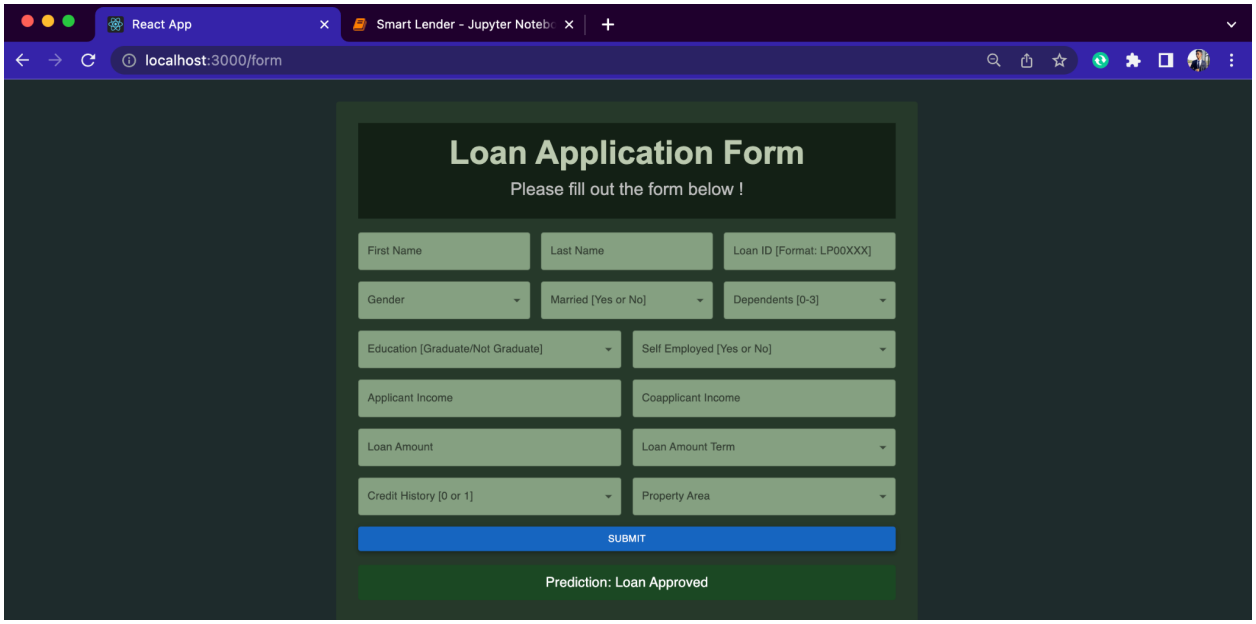


- Testimonial Section
    - Slider window with 3 sections
    - Hover animation effect with pretty styling over each card that indicates the reviews of the user



- Form Page which can be accessed by clicking on the Button





Custom prediction labels with appropriate styling for both Approved and Rejected Status

7 ADVANTAGES & DISADVANTAGES

Advantages:

- The system leverages machine learning techniques and compares various classification models to achieve high accuracy -96% with XGBoost.

- By automating the loan eligibility assessment, the system eliminates the need for manual evaluation, reducing processing time and improving overall efficiency. It enables users to quickly determine their loan eligibility without lengthy paperwork or delays.

- User-Friendly Interface built using modern frameworks like React, Formik, and Yup, allowing borrowers to easily input their details, navigate through the application, and receive clear loan eligibility results.

- The system follows a modular architecture, allowing for scalability and easy integration with additional features or enhancements in the future. This flexibility ensures that the system can adapt to changing business requirements and accommodate potential growth.

Disadvantages:

- The system's accuracy heavily relies on the quality and relevance of historical loan data used for training the machine learning models. If the historical data is outdated, biased,

or not representative of the target population, it may affect the system's accuracy and reliability.
- Potential Bias in Predictions
- It may be challenging to explain the specific factors or criteria that contribute to a loan eligibility decision made by the model. This lack of transparency may raise concerns among users and stakeholders who require clear explanations for decision-making.

- The Lendwise Smart Lender System heavily relies on technology infrastructure, including servers, databases, APIs, and frameworks.
- Any technical issues, system downtime, or cybersecurity vulnerabilities could impact the availability and functionality of the system, potentially disrupting loan application processing.

8 APPLICATIONS

- Banks and financial institutions can leverage the system to streamline their loan application processes. It enables them to efficiently evaluate loan eligibility, reduce manual work, and expedite decision-making.
- Peer-to-peer lending platforms and online lenders can integrate the Lendwise system to automate and improve their loan eligibility assessments.
- Credit unions can utilize the Lendwise system to simplify their loan approval processes.
- Mortgage companies can incorporate the Lendwise system to streamline mortgage application and approval processes. It assists in evaluating borrower eligibility based on predefined criteria, ensuring a smoother and faster mortgage lending experience.
- Consumer finance companies offering personal loans, auto loans, or credit services can employ the Lendwise system to automate loan eligibility assessments. This enables them to provide customers with quick loan decisions, enhance risk management, and improve operational efficiency.
- Government agencies offering loan programs, such as small business loans or student loans, can utilize the Lendwise system to simplify the application and eligibility determination processes.

9 CONCLUSION

The development of the Lendwise Smart Lender System has resulted in a revolutionary solution for loan eligibility assessment. By leveraging machine learning techniques, the system achieves a remarkable accuracy of 96% using the XGBoost model and also indicates the accuracies of various other popular algorithms. The use of modern technologies, such as React for the frontend and Flask for the backend, ensures a robust and user-friendly system. With its ability to automate and streamline the loan application process, the Lendwise system offers advantages such as improved efficiency, reduced errors, and accelerated decision-making for financial institutions. Its applicability extends to various sectors, including traditional banks, online lending

platforms, microfinance institutions, and government loan programs. Overall, the Lendwise Smart Lender System presents a cutting-edge solution that enhances loan assessment accuracy and facilitates a seamless borrowing experience.

10 FUTURE SCOPE

- Integration of additional data sources for improved accuracy
- Continuous model training and retraining
- Advanced feature engineering techniques
- Integration of different loan types
- Enhanced user interface and experience
- Integration with external systems for verification
- Compliance and security enhancements to ensure data privacy
- Scalability to handle increasing user demands
- Incorporation of feedback mechanisms for system improvement
- Integration with credit scoring models for comprehensive risk assessment
- Development of mobile applications for easy access and convenience
- Localization and internationalization to cater to diverse markets
- Collaboration with financial institutions for data sharing and analysis
- Integration with financial APIs for real-time data retrieval
- Incorporation of natural language processing for automated document analysis
- Integration with cloud-based infrastructure for scalability and cost efficiency.

11 BIBLIOGRAPHY

- "Hands-On Machine Learning with Scikit-Learn and TensorFlow" by Aurélien Géron
- "Python Machine Learning" by Sebastian Raschka and Vahid Mirjalili
- "Flask Web Development with Python Tutorial" on Flask's official documentation
- "React Documentation" on the official React website
- "Formik Documentation" on the official Formik website
- "Yup Documentation" on the official Yup GitHub repository
- "Machine Learning Mastery" by Jason Brownlee
- "Towards Data Science" for articles and tutorials on machine learning and data science

APPENDIX A. Source Code Attach the code for the solution built.

Front End (using react)

Home page

```
import {Testimonials} from "./Testimonials";
import {Banner} from "./Banner";
import {About} from "./About";
import {NavBar} from "./NavBar";

function HomePage() {
return (
```

```
<div className="App">
<NavBar></NavBar>
<Banner></Banner>
<About></About>
<Testimonials></Testimonials>
</div>
);
}
export default HomePage;
```

Form Page

```
import React, { useState } from "react";
import { Container, Grid, Typography } from "@mui/material";
import { Formik, Form } from "formik";
import * as Yup from "yup";
import Select from "./FormUI/Select";
import TextField from "./FormUI/TextField";
import Button from "./FormUI/Button";
import "./FormPage.css";

const INITIAL_FORM_STATE = {
firstName: "",
lastName: "",
loanID: "",
Gender: "",
Married: "",
Dependents: "",
Education: "",
Self_Employed: "",
ApplicantIncome: "",
CoapplicantIncome: "",
LoanAmount: "",
Loan_Amount_Term: "",
Credit_History: "",
Property_Area: "",
};

const FORM_VALIDATION = Yup.object().shape({
firstName: Yup.string().required("Required Field"),
lastName: Yup.string().required("Required Field"),
loanID: Yup.string().required("Required Field"),
Gender: Yup.string().required("Required"),
```

```
Married: Yup.string().required("Required"),
Dependents: Yup.string().required("Required"),
Education: Yup.string().required("Required"),
Self_Employed: Yup.string().required("Required"),
ApplicantIncome: Yup.string().required("Required"),
CoapplicantIncome: Yup.string().required("Required"),
LoanAmount: Yup.string().required("Required"),
Loan_Amount_Term: Yup.string().required("Required"),
Credit_History: Yup.string().required("Required"),
Property_Area: Yup.string().required("Required"),
});

export const FormPage = () => {
const [prediction, setPrediction] = useState(null);

const handleSubmit = async (values, { resetForm }) => {
console.log("Form data:", values);
try {
const response = await fetch("http://localhost:8000/form", {
method: "POST",
headers: {
"Content-Type": "application/json",
},
body: JSON.stringify(values),
});

if (response.ok) {
const data = await response.json();
const predictions = data.predictions;
setPrediction(predictions[0]);
resetForm();
} else {
console.log("Form submission failed");
}
} catch (error) {
console.log("An error occurred", error);
}
};

return (
<Grid container>
<Grid item xs={12}>
```

```
<Container maxWidth="md">
<div className="form-wrapper">
<div className="boxbanner">
<h1 className="boxbanner-title">Loan Application Form</h1>
<p className="boxbanner-subtitle">Please fill out the form below !</p>
</div>
<Formik
initialValues={INITIAL_FORM_STATE}
validationSchema={FORM_VALIDATION}
onSubmit={handleSubmit}
>
<Form>
<Grid container spacing={2}>
<Grid item xs={4}>
<TextField
name="firstName"
label="First Name"
className="input-field"
/>
</Grid>
<Grid item xs={4}>
<TextField
name="lastName"
label="Last Name"
className="input-field"
/>
</Grid>
<Grid item xs={4}>
<TextField
name="loanID"
label="Loan ID [Format: LP00XXX]"
className="input-field"
/>
</Grid>

<Grid item xs={4}>
<Select
name="Gender"
label="Gender"
options={["Male", "Female"]}
className="input-field"
/>
```

```
</Grid>
<Grid item xs={4}>
<Select
name="Married"
label="Married [Yes or No]"
options={["Yes", "No"]}
className="input-field"
/>
</Grid>
<Grid item xs={4}>
<Select
name="Dependents"
label="Dependents [0-3]"
options={[0, 1, 2, 3]}
className="input-field"
/>
</Grid>

<Grid item xs={6}>
<Select
name="Education"
label="Education [Graduate/Not Graduate]"
options={["Graduate", "Not Graduate"]}
className="input-field"
/>
</Grid>

<Grid item xs={6}>
<Select
name="Self_Employed"
label="Self Employed [Yes or No]"
options={["Yes", "No"]}
className="input-field"
/>
</Grid>

<Grid item xs={6}>
<TextField
name="ApplicantIncome"
label="Applicant Income"
className="input-field"
/>
```

```
</Grid>

<Grid item xs={6}>
<TextField
name="CoapplicantIncome"
label="Coapplicant Income"
className="input-field"
/>
</Grid>

<Grid item xs={6}>
<TextField
name="LoanAmount"
label="Loan Amount"
className="input-field"
/>
</Grid>

<Grid item xs={6}>
<Select
name="Loan_Amount_Term"
label="Loan Amount Term"
options={[180, 360, 480]}
className="input-field"
/>
</Grid>

<Grid item xs={6}>
<Select
name="Credit_History"
label="Credit History [0 or 1]"
options={[0, 1]}
className="input-field"
/>
</Grid>

<Grid item xs={6}>
<Select
name="Property_Area"
label="Property Area"
options={["Rural", "Semiurban", "Urban"]}
className="input-field"
```

```
/>
</Grid>

<Grid item xs={12}>
<Button type="submit" className="submit-button">
Submit
</Button>
</Grid>
</Grid>
</Form>
</Formik>
{prediction !== null && (
<div
className={`prediction-container ${
prediction === 0 ? "rejected" : "approved"
}`}
>
<Typography variant="h6" className="prediction-text">
Prediction:{" "}
{prediction === 0 ? "Loan Rejected" : "Loan Approved"}
</Typography>
</div>
)}
</div>
</Container>
</Grid>
</Grid>
);
};
```

There are various other components that were rendered for the making of this page such as: About, Banner, Testimonials,TestimonialCard and Form Page components that abstracted the functionality of the form fields.
Adding these would elongate the document and can be referred to from the github repo (as there are not essential)

Backend (using flask)

```
from flask import Flask, jsonify, request
from flask_cors import CORS
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
```

```
from joblib import load

app = Flask(__name__)
CORS(app)

model = load('knn_model.joblib')

scaler = load('scaler.joblib')

@app.route('/form', methods=['POST'])
def form_predict():
data = request.get_json(force=True)
df = pd.DataFrame(data, index=[0])
le = LabelEncoder()
df["Gender"] = le.fit_transform(df["Gender"])
df["Married"] = le.fit_transform(df["Married"])
df["Education"] = le.fit_transform(df["Education"])
df["Self_Employed"] = le.fit_transform(df["Self_Employed"])
df["Property_Area"] = le.fit_transform(df["Property_Area"])
df["Dependents"] = df["Dependents"].astype(float)
df.drop(["loanID", "lastName", "firstName"], axis=1, inplace=True)

df["ApplicantIncome"] = pd.to_numeric(df["ApplicantIncome"])
df["CoapplicantIncome"] = pd.to_numeric(df["CoapplicantIncome"])
df["LoanAmount"] = pd.to_numeric(df["LoanAmount"])
df["Loan_Amount_Term"] = pd.to_numeric(df["Loan_Amount_Term"])
df["Credit_History"] = pd.to_numeric(df["Credit_History"])

df_scaled = df.copy()
numerical_columns = ["ApplicantIncome", "CoapplicantIncome", "LoanAmount",
"Loan_Amount_Term"]
df_scaled[numerical_columns] = scaler.transform(df[numerical_columns])

predictions = model.predict(df_scaled)
response = jsonify({"predictions": predictions.tolist()})
return response

if __name__ == "__main__":
app.run(port=8000, debug=True)
```