WORLD HAPPINESS ANALYSIS USING PYSPARK

A MINI PROJECT REPORT

BHUVANIKA S (9517202109011)

RAJAKUMARI S (9517202109042)

SUJI S (9517202109051)

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

MEPCO SCHLENK ENGINEERING COLLEGE, SIVAKASI

ANNA UNIVERSITY: CHENNAI 600 025

NOVEMBER 2024

MEPCO SCHLENK ENGINEERING COLLEGE, SIVAKASI

AUTONOMOUS

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATASCIENCE

BONAFIDE CERTIFICATE

This is to certify that it is the bonafide work of **BHUVANIKA S (9517202109011) , RAJAKUMARI S (9517202109042) , SUJI S (9517202109051)** for the mini project titled **WORLD HAPPINESS ANALYSIS USING PYSPARK** in 19AD701 – Big Data Analytics And Technologies during the seventh semester July 2024 – November 2024 under my supervision.

SIGNATURE

Dr.S. Shiny M.E.,Ph.D

Assistant Professor(SG)

Artificial Intelligence and Data Science

Mepco Schlenk Engineering College

Sivakasi – 626 005

Virudhunagar District.

SIGNATURE

Dr.J.AngelaJennifa Sujana, M.E.,Ph.D

Professor& Head

Artificial Intelligence and Data Science

Mepco Schlenk Engineering College,

Sivakasi – 626 005.

Virudhunagar District

ABSTRACT

This project presents a web-based application designed for regression analysis and data visualization using machine learning algorithms, developed with the Streamlit framework. The application enables users to upload datasets, select relevant features and target variables, and choose from a variety of regression models, including Linear Regression, Random Forest, XGBoost, and more. The primary objective is to provide a user-friendly interface that facilitates the exploration and evaluation of different modeling techniques without requiring extensive programming knowledge.

Key functionalities of the application include data preprocessing, imputation of missing values, train-test splitting, and the calculation of essential performance metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R^2 score. Users can visualize results through line charts and scatter plots, enhancing their understanding of the relationships between actual and predicted values.

Additionally, the application integrates a SQL database containing the World Happiness Report, allowing users to execute queries and retrieve insights about happiness scores across various countries and factors. This functionality enables a deeper analysis of the social and economic determinants of happiness, contributing to informed discussions and data-driven decision-making.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENT

First and foremost, we **praise and thank "The Almighty",** the lord of all creations, who by his abundant grace has sustained us and helped us to work on this project successfully.

We really find unique pleasure and immense gratitude in thanking our respected management members**,** who is the backbone of our college.

A deep bouquet of thanks to respected Principal **Dr.S.Arivazhagan M.E.,Ph.D.,** for having provided the facilities required for our mini project.

We sincerely thank our Head of the Department **Dr. J. Angela Jennifa Sujana M.E.,Ph.D.,** Professor & Head, Department of Artificial Intelligence and Data Science, for her guidance and support throughout the mini project .

We also thank our guide **Dr.S.Shiny M.E.,Ph.D.,** Assistant Professor(Senior Grade), Department of Artificial Intelligence and Data Science, for her valuable guidance and it is great privilege to express our gratitude to her.

We extremely thank our project coordinator **Mrs. L.Prasika.,M.E,(Ph.D),** Assistant Professor, Department of Artificial Intelligence and Data Science, who inspired us and supported us throughout the mini project.

We extend our heartfelt thanks and profound gratitude to all the faculty members of Artificial Intelligence and Data Science department for their kind help during our mini project work.

We also thank our parents and our friends who had been providing us with constant support during the course of the mini project work .

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

This project combines machine learning regression models and data analysis in a user-friendly web application built with Streamlit. The application allows users to upload their own datasets and select various machine learning models to train and evaluate. It supports multiple regression algorithms like Linear Regression, Random Forest, XGBoost, and others, enabling users to compare model performance based on metrics such as MAE, MSE, RMSE, and R^2 score. Users can also visualize the predictions through line charts and scatter plots, offering insights into the models' performance on actual versus predicted values.

In addition to machine learning, the project includes an analytical component based on the World Happiness Report. Using SQL queries, the app extracts meaningful insights from the dataset, such as identifying the happiest and least happy countries, analyzing happiness scores by region, and examining key factors like GDP, social support, and freedom. These queries allow users to explore patterns and trends in global happiness and related socioeconomic indicators.

By merging machine learning with SQL-based data analysis, the project provides an interactive platform that simplifies both predictive modeling and dataset exploration. This makes it suitable for a wide range of applications, from model evaluation to insightful data-driven decision-making.

1.2 OBJECTIVE

The goal of this project is to develop an accessible, web-based application that streamlines regression analysis and data-driven insights using machine learning models. The application is designed to simplify machine learning workflows by providing a user-friendly interface through Streamlit, enabling users to upload datasets, select features, and apply various regression models without requiring extensive programming knowledge. It aims to offer the ability to compare multiple regression algorithms, such as Linear Regression, Random Forest, and XGBoost, while generating key performance metrics like MAE, MSE, RMSE, and R^2 score.

Additionally, the project emphasizes the importance of data visualization by incorporating tools like line charts and scatter plots to help users interpret model outcomes by comparing actual and predicted values. A significant feature of the application is its integration with a SQL database (e.g., the World Happiness Report), allowing users to execute queries, extract insights, and analyze patterns related to global happiness and associated factors. Ultimately, this project aims to empower users to make data-driven decisions by leveraging machine learning models, data analysis techniques, and visualizations in a seamless, intuitive platform.

CHAPTER 2

PROBLEM STATEMENT

2.1 FLOW DIAGRAM

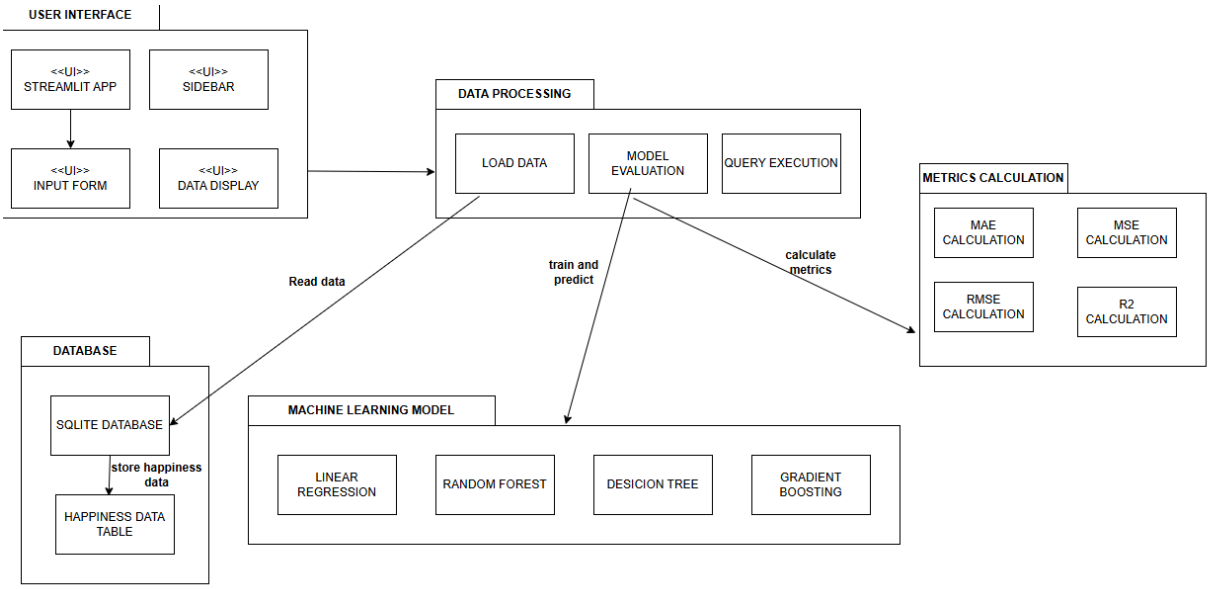

Figure 2.1-Flow diagram

represents the architecture of a machine learning regression platform built with Streamlit, integrating various components for data processing and analysis. The User Interface (UI), created using Streamlit, allows users to interact with the app through input forms and data display. The Data Processing block handles loading data, evaluating models, and executing SQL queries. It interacts with the Machine Learning Model section, where users can choose from models like Linear Regression, Random Forest, Decision Tree, and Gradient Boosting for training and prediction. The Metrics Calculation unit computes essential performance metrics (MAE, MSE, RMSE, R^2) for model evaluation. The Database block, powered by SQLite, stores the happiness data for analysis. Together, these components ensure that data is read, processed, and analyzed efficiently within the platform.

2.2 WORK FLOW

1. User Interaction with the Web Application

- Users interact with the application through the Streamlit interface.
- When users open the application, they are presented with an intuitive interface where they can upload their dataset, select features, choose a regression model, and visualize the results. The interface simplifies complex tasks, making machine learning workflows accessible without requiring users to write code.

2. Dataset Upload

- Users upload a CSV dataset.
- The system allows users to upload their datasets in CSV format. The data is loaded and previewed within the interface, allowing the user to verify the dataset's content. This is a crucial step as the analysis and model training depend on the quality and structure of the uploaded data.

3. Feature and Target Selection

- Users select the target variable (what they want to predict) and features (input variables).
- The application dynamically displays the columns from the uploaded dataset, prompting users to select the target variable (dependent variable) and the feature variables (independent variables). This allows users to specify which part of the dataset will be used to train the machine learning models.

4. **Data Preprocessing**

- Data cleaning and preprocessing are done automatically.
- After selecting the features and target, the system preprocesses the data. It handles missing values using techniques such as mean imputation (filling missing values with the column mean), converts categorical variables into numerical representations using one-hot encoding (if necessary), and ensures the data is in a format suitable for machine learning models.

5. Train-Test Split

- The dataset is split into training and testing sets.

- The data is divided into training (80%) and testing (20%) sets. The training set is used to train the regression models, while the testing set is reserved for evaluating the model's performance. This ensures that the model's accuracy is measured on data it hasn't seen before, giving a more realistic assessment of its predictive power.

6. Model Selection

- Users choose a regression model from a list of options.
- The application provides users with a variety of regression models to choose from, including:
    - Linear Regression
    - Random Forest
    - Decision Tree

    Gradient Boosting

Each model has its strengths, and users can experiment with different models to see which works best for their data.

7. Model Training

- The selected model is trained on the training dataset.
- Once the model is selected, the application trains the model using the training data. Training involves the model learning from the data to identify patterns or relationships between the features and the target variable. The model then optimizes its internal parameters to minimize error and improve its predictions.

8. Model Prediction

- The trained model is used to make predictions on the test set.
- After training, the model is tested on the unseen data (test set). It generates predictions based on the input features from the test set. This step helps assess how well the model generalizes to new data.

9. Model Evaluation

- Performance metrics are calculated to evaluate the model.
- The application calculates several performance metrics to measure the accuracy of the predictions:
    - Mean Absolute Error (MAE): Measures the average absolute difference between predicted and actual values.
    - Mean Squared Error (MSE): Measures the average squared difference between predicted and actual values.
    - Root Mean Squared Error (RMSE): The square root of MSE, providing an error estimate in the same units as the target variable.
    - R^2 Score: Indicates how well the model fits the data, with 1 being a perfect fit and 0 meaning no fit.

These metrics help users understand the effectiveness of the model and compare different models' performance.

10. Visualization of Results

- Visual representations of actual vs. predicted values are generated.
- The application provides several visualization tools to help users interpret the results:
    - Line Charts: Compare the actual and predicted values over the data points.
    - Scatter Plots: Show the relationship between actual and predicted values. Ideally, the points should align closely to the diagonal line, indicating accurate predictions.

Visualization allows users to visually assess how well the model is performing and identify any trends or patterns in the data.

11. SQL Database Querying (Happiness Data)

- Users can run SQL queries to analyze insights from the World Happiness Report.

- In addition to regression analysis, the application connects to a SQLite database that contains the World Happiness Report data. Users can run predefined SQL queries to analyze various aspects of global happiness, such as identifying the happiest countries, comparing GDP and happiness scores, and understanding social support metrics. These insights can be used to augment decision-making or support broader data-driven analyses.

12. User Output and Insights

- The application presents the final model results and data insights to the user.
- After training and evaluating the models, users receive a comprehensive summary of the performance metrics, visualizations, and any relevant insights from the database. This final step provides the user with the results necessary to interpret the data and make informed decisions based on the analysis.

13. Comparison and Decision Making

- Users compare models and interpret results.
- The final step involves the user comparing different models and deciding which model provides the best performance based on the calculated metrics and visual results. The application allows users to rerun the analysis with different models or datasets, enabling iterative refinement and deeper exploration.

2.3 MODULES REQUIRED

2.3.1 How Streamlit is used:

- **Frontend User Interface:** Streamlit is used to create the interactive web interface for the project.
- It allows users to upload datasets, choose features, select models, and view visualizations and results directly in the browser.

- **Uploading Data:** The st.file_uploader() method is used to let users upload CSV datasets, which are then read into a Pandas DataFrame for processing.
- **Displaying Data and Results:** Streamlit widgets like st.write(), st.selectbox(), and st.multiselect() allow users to preview the dataset and select the target and features for the model.
- **Model Selection and Training:** Streamlit's st.selectbox() allows users to choose from multiple machine learning models for training and evaluation.
- **Visualizations:** Streamlit is integrated with Matplotlib to plot and display visualizations like scatter plots and line charts of actual vs predicted values.

2.3.2 How Scikit-learn is used:

- **Model Initialization:** The project uses multiple regression models from Scikit-learn, such as RandomForestRegressor, LinearRegression, and SVR. Each model is selected based on user input and trained on the dataset.
- **Data Preprocessing:** SimpleImputer is used to handle missing data by imputing missing values with the mean or another specified strategy.
- **Model Evaluation:** Scikit-learn metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R^2 Score are used to evaluate model performance.
- **Train-Test Split:** The train_test_split function is used to split the data into training and testing sets, ensuring the model is evaluated on unseen data.

2.3.3 How Pandas is used:

- **Loading and Handling Data:** Pandas is used to load the CSV dataset into a DataFrame, which is then used for model training and evaluation.
- **Data Preprocessing:** The get_dummies() function from Pandas is used to handle categorical features by encoding them as numeric columns

- **Data Manipulation:** Pandas allows for quick data inspection and manipulation, such as previewing data with.head() and selecting relevant features.

2.3.4 How Matplotlib is used:

- **Visualizing Results:** Matplotlib is used to create scatter plots and line charts that show the actual vs predicted values for the regression models.
- **Plotting Model Performance:** After the model evaluation, Matplotlib plots are integrated into Streamlit for easy visualization of results, helping users analyze the model's performance visually.

2.3.5 How SQLite3 is used:

- **Database Connectivity:** The project uses SQLite3 to connect to a database containing the World Happiness Report data.
- **Query Execution:** SQL queries are executed to fetch data from the database based on user input, such as retrieving the happiest country or calculating average happiness scores.
- **Data Analysis:** After querying, the results are displayed in Streamlit for the user to analyze and compare with the regression model results.

2.4 ABOUT DATASET

The World Happiness Report dataset ranks countries based on their happiness levels and analyzes various factors that contribute to these happiness scores. Each row in the dataset represents a country, while the columns provide a range of information that helps explain the differences in happiness levels across countries. Key columns include the Ladder Score, which is the main happiness ranking (on a scale of 0 to 10) for each country. The dataset also includes upper and lower bounds (called Upper Whisker and Lower Whisker) that give a confidence interval for the score. Additionally, several socio-economic factors are measured, such as the

Log GDP per Capita, which represents the wealth or prosperity of the country, and Social Support, which indicates the strength of the social networks available to people. Healthy Life Expectancy shows how long people in a country are expected to live in good health, while Freedom to Make Life Choices reflects how free people feel to make decisions about their own lives. Other columns, such as Generosity and Perception of Corruption, measure the general levels of charitable behavior and perceived corruption in each country, respectively. The column Dystopia + Residual represents a baseline for the lowest possible national happiness score, adjusted with residual errors from the model.

Country n	Regional i	Ladder sc	upperwhi	lowerwhi	Log GDP p	Social sup	Healthy li	Freedom t	Generosit	Perceptio	Dystopia + residua
Finland	Western E	7.741	7.815	7.667	1.844	1.572	0.695	0.859	0.142	0.546	2.082
Denmark	Western E	7.583	7.665	7.5	1.908	1.52	0.699	0.823	0.204	0.548	1.881
Iceland	Western E	7.525	7.618	7.433	1.881	1.617	0.718	0.819	0.258	0.182	2.05
Sweden	Western E	7.344	7.422	7.267	1.878	1.501	0.724	0.838	0.221	0.524	1.658
Israel	Middle Ea	7.341	7.405	7.277	1.803	1.513	0.74	0.641	0.153	0.193	2.298
Netherlan	Western E	7.319	7.383	7.256	1.901	1.462	0.706	0.725	0.247	0.372	1.906
Norway	Western E	7.302	7.389	7.215	1.952	1.517	0.704	0.835	0.224	0.484	1.586
Luxembou	Western E	7.122	7.213	7.031	2.141	1.355	0.708	0.801	0.146	0.432	1.54
Switzerlar	Western E	7.06	7.147	6.973	1.97	1.425	0.747	0.759	0.173	0.498	1.488
Australia	North Am	7.057	7.141	6.973	1.854	1.461	0.692	0.756	0.225	0.323	1.745
New Zeal	North Am	7.029	7.105	6.954	1.81	1.527	0.673	0.746	0.226	0.48	1.567
Costa Rica	Latin Ame	6.955	7.051	6.86	1.561	1.373	0.661	0.797	0.109	0.123	2.333
Kuwait	Middle Ea	6.951	7.06	6.843	1.845	1.364	0.661	0.827	0.2	0.172	1.884
Austria	Western E	6.905	6.986	6.824	1.885	1.336	0.696	0.703	0.214	0.305	1.766
Canada	North Am	6.9	6.984	6.815	1.84	1.459	0.701	0.73	0.23	0.368	1.572
Belgium	Western E	6.894	6.961	6.827	1.868	1.44	0.69	0.729	0.17	0.311	1.686
Ireland	Western E	6.838	6.927	6.749	2.129	1.39	0.7	0.758	0.205	0.418	1.239
Czechia	Central an	6.822	6.903	6.741	1.783	1.511	0.638	0.787	0.177	0.068	1.858
Lithuania	Central an	6.818	6.896	6.739	1.766	1.454	0.598	0.533	0.044	0.116	2.307
United Kir	Western E	6.749	6.833	6.665	1.822	1.326	0.672	0.713	0.267	0.351	1.598
Slovenia	Central an	6.743	6.843	6.643	1.786	1.502	0.695	0.789	0.157	0.131	1.683
United Ar	Middle Ea	6.733	6.823	6.643	1.983	1.164	0.563	0.815	0.209	0.258	1.741
United Sta	North Am	6.725	6.818	6.631	1.939	1.392	0.542	0.586	0.223	0.169	1.873
Germany	Western E	6.719	6.815	6.622	1.871	1.39	0.702	0.7	0.174	0.368	1.513
Mexico	Latin Ame	6.678	6.781	6.575	1.521	1.241	0.544	0.722	0.086	0.127	2.437
Uruguay	Latin Ame	6.611	6.696	6.527	1.596	1.431	0.592	0.775	0.106	0.22	1.891
France	Western E	6.609	6.685	6.533	1.818	1.348	0.727	0.65	0.112	0.281	1.672
Saudi Aral	Middle Ea	6.594	6.707	6.48	1.842	1.361	0.511	0.787	0.114	0.188	1.79

Figure 2.2-Dataset

CHAPTER 3

PROPOSED METHOD

3.1 SOURCE CODE

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import streamlit as st
from pyspark.ml import Pipeline
from pyspark.ml.evaluation import RegressionEvaluator
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.regression import (DecisionTreeRegressor, GBTRegressor
 LinearRegression, RandomForestRegressor)
from pyspark.sql import SparkSession
from pyspark.sql.functions import avg
spark = SparkSession.builder \
    .appName("World Happiness Report Analysis") \
    .getOrCreate()
def load_data()
    df = spark.read.csv("B:/Documents/MINI PROJECT REPORTS/SEM 7 MINI
PROJECT/BD/World-Happiness-main/World-Happiness-main/World-happiness-
report-2024.csv", header=True, inferSchema=True)
    return df
def evaluate_model(model, X_train, y_train, X_test, y_test):
    model.fit(X_train, y_train)# Trains the model using training data.
    predictions = model.transform(X_test).

evaluator=RegressionEvaluator(labelCol="label",predictionCol="predicti
on", metricName="rmse”)
    rmse = evaluator.evaluate(predictions)
```

```
mae_evaluator=RegressionEvaluator(labelCol="label",predictionCol="pred
iction", metricName="mae")
    mae = mae_evaluator.evaluate(predictions)
    mse_evaluator        =        RegressionEvaluator(labelCol="label",
predictionCol="prediction", metricName="mse")
    mse = mse_evaluator.evaluate(predictions)
    r2_evaluator        =        RegressionEvaluator(labelCol="label",
predictionCol="prediction", metricName="r2")
    r2 = r2_evaluator.evaluate(predictions)
    return mae, mse, rmse, r2
st.title("World Happiness Report Analysis")
st.sidebar.title("Regression Model Comparison")
models = {#A dictionary that maps model names to their respective
instances
    'Linear Regression': LinearRegression(),
    'Random Forest': RandomForestRegressor(),
    'Decision Tree': DecisionTreeRegressor(),
    'Gradient Boosting': GBTRegressor(),
}
selected_models     =     st.sidebar.multiselect("Select     Models",
list(models.keys()), default=["Linear Regression"])
data = load_data()
if data.count() > 0:#Counts the number of rows in the DataFrame to
check if data was loaded successfully.
    st.write("Dataset Loaded Successfully")
        if st.checkbox('Show Dataset'):
        st.write(data.toPandas())  # Convert to Pandas for display
    feature_columns = st.multiselect("Select Features", data.columns)
    target_column    =    st.selectbox("Select    Target    Variable",
data.columns)
```

```
    if feature_columns and target_column:
        assembler = VectorAssembler(inputCols=feature_columns, outputCol="features")#combines multiple columns into a single vector column.
        data = assembler.transform(data)#Transforms the DataFrame to include a "features" column
        data = data.select("features", target_column).withColumnRenamed(target_column, "label")
        train_data, test_data = data.randomSplit([0.8, 0.2], seed=42)
        metrics_df = pd.DataFrame(columns=["Model", "MAE", "MSE", "RMSE", "R²"])#to store evaluation metrics for each model
        for model_name in selected_models:
            model = models[model_name]
            mae, mse, rmse, r2 = evaluate_model(model, train_data, train_data, test_data, test_data)
            new_row = pd.DataFrame({
                "Model": [model_name],
                "MAE": [mae],
                "MSE": [mse],
                "RMSE": [rmse],
                "R²": [r2]
            })
            metrics_df = pd.concat([metrics_df, new_row], ignore_index=True)# Concatenates the new row to the metrics DataFrame, ignoring the index to keep it clean.
        st.write("Model Performance:")
        st.dataframe(metrics_df)
        st.bar_chart(metrics_df.set_index("Model")[["MAE", "MSE", "RMSE"]])
        if st.sidebar.button("View Comparison Plot"):
            st.subheader("Comparison of Model Performance Metrics")
```

```
            fig, axs = plt.subplots(4, 1, figsize=(10, 12))
            axs[0].barh(metrics_df["Model"],        metrics_df["MAE"],
color='orange')
            axs[0].set_title('Mean Absolute Error (MAE)')
            axs[0].set_xlim(0, metrics_df["MAE"].max() + 0.1)
         axs[1].barh(metrics_df["Model"],          metrics_df["RMSE"],
color='blue')
            axs[1].set_title('Root Mean Squared Error (RMSE)')
            axs[1].set_xlim(0, metrics_df["RMSE"].max() + 0.1)
         axs[2].barh(metrics_df["Model"],           metrics_df["R²"],
color='green')
            axs[2].set_title('R² Score')
            axs[2].set_xlim(0, 1)
         axs[3].barh(metrics_df["Model"],          metrics_df["MSE"],
color='red')
            axs[3].set_title('Mean Squared Error (MSE)')
            axs[3].set_xlim(0, metrics_df["MSE"].max() + 0.1)
         for ax in axs:
               ax.set_xlabel('Score')
               ax.set_ylabel('Models')
         plt.tight_layout()
            st.pyplot(fig)  # Display the plot
        st.subheader("Enter Feature Values for Prediction")
        input_data = {}
     for feature in feature_columns:
            input_data[feature]  =  st.number_input(f"Enter  value  for
{feature}:", value=0.0, step=0.01)
       input_df = pd.DataFrame([input_data])
        input_spark_df = spark.  createDataFrame (input_df)
     input_spark_df = assembler.transform(input_spark_df)
```

```
        # Predict using the first selected model if available
        if selected_models:
            model = models[selected_models[0]]
            predicted_score = model.transform(input_spark_df)

            # Get the prediction
            predicted_value =
predicted_score.collect()[0]["prediction"]

            # Determine the happiness category based on the predicted
score
            if predicted_value >= 6.0:   # Adjust this threshold as
needed
                happiness_status = "😀"  # High happiness
                emoji_size = "<h1 style='font-size:50px;'>😀</h1>"   #
Big emoji
                status_text = "High Happiness"
            elif predicted_value >= 4.0:   # Adjust this threshold as
needed
                happiness_status = "😳"  # Average happiness
                emoji_size = "<h1 style='font-size:50px;'>😳</h1>"   #
Big emoji
                status_text = "Average Happiness"
            else:
                happiness_status = "😭"  # Low happiness
                emoji_size = "<h1 style='font-size:50px;'>😭</h1>"   #
Big emoji
                status_text = "Low Happiness"

            # Display the predicted happiness score and status
```

```
            st.write(f"Predicted Happiness Score: {predicted_value:.2f}")
            st.markdown(emoji_size, unsafe_allow_html=True)  # Display the emoji
            st.write(status_text)  # Display the corresponding status text
else:
    st.write("No data available. Please load the dataset.")

# Placeholder for metrics
st.subheader("Model Performance Metrics")

# Queries
queries = {
    "Happiest Country": "SELECT Country_name, Ladder_score FROM HappinessData12 ORDER BY Ladder_score DESC LIMIT 1",
    "Average Happiest Country": "SELECT AVG(Ladder_score) AS AverageHappiness FROM HappinessData12",
    "Least Happiest Country": "SELECT Country_name, Ladder_score FROM HappinessData12 ORDER BY Ladder_score ASC LIMIT 1",
    "Top 10 Highest GDP Countries": "SELECT Country_name, Log_GDP_per_capita FROM HappinessData12 ORDER BY Log_GDP_per_capita DESC LIMIT 10",
    "Top 10 Lowest GDP Countries": "SELECT Country_name, Log_GDP_per_capita FROM HappinessData12 ORDER BY Log_GDP_per_capita ASC LIMIT 10",
    "Average Happiness Score by Region": "SELECT Regional_indicator, AVG(Ladder_score) AS AverageHappiness FROM HappinessData12 GROUP BY Regional_indicator ORDER BY AverageHappiness DESC",
```

```
    "Top 10 Countries with Best Social Support": "SELECT Country_name, Social_support FROM HappinessData12 ORDER BY Social_support DESC LIMIT 10",
    "Top 10 Countries with Highest Freedom": "SELECT Country_name, Freedom_to_make_life_choices FROM HappinessData12 ORDER BY Freedom_to_make_life_choices DESC LIMIT 10",
    "Top 10 Countries with Lowest Corruption": "SELECT Country_name, Perceptions_of_corruption FROM HappinessData12 ORDER BY Perceptions_of_corruption ASC LIMIT 10",
    "Top 5 Countries for Generosity": "SELECT Country_name, Generosity FROM HappinessData12 ORDER BY Generosity DESC LIMIT 5",
    "Countries Above Average Happiness": "SELECT Country_name, Ladder_score FROM HappinessData12 WHERE Ladder_score > (SELECT AVG(Ladder_score) FROM HappinessData12)"
}

# Create sidebar for queries
st.sidebar.header("Select a Query")
query_selection = st.sidebar.selectbox("Choose a query to display results:", list(queries.keys()))

# Execute the selected query and display results
if query_selection:
    sql_query = queries[query_selection]
    result_df = spark.sql(sql_query).toPandas()  # Execute query and convert to Pandas DataFrame

    # Display the result in the Streamlit app
    st.subheader(query_selection)
    st.dataframe(result_df)
```

Insights

Model Comparison and Performance: Different regression models yield different levels of accuracy based on the dataset.Users can easily compare models like Linear Regression, Random Forest etc., using metrics such as MAE, MSE, and R^2.

Feature Selection Impact: The choice of input features affects the model's accuracy. Users can interactively select and deselect features, optimizing the model for better predictions. Visualizing Actual vs. Predicted Values: Line charts and scatter plots allow users to visually assess model performance .Close alignment between actual and predicted values indicates good model performance.

Automatic Handling of Missing Data: Missing values are handled using mean imputation, ensuring the model can still function with incomplete data. Reduces the need for manual data cleaning

World Happiness Report Insights: SQL queries offer insights like the happiest/least happy countries and the relationship between GDP, social support, and happiness. Users can identify global happiness trends and key influencing factors.

Adaptability to Different Datasets :The project can be applied to any dataset by allowing users to upload data and select target variables and features. It is versatile across multiple domains (finance, healthcare, etc.).

3.2 OUTPUTS AND VISUALIZATION

Figure 3.1-Dataset loaded

Figure 3.1 shows the user interface of the *World Happiness Report Analysis* platform. On the left, users can select a regression model (e.g., Linear Regression) and a predefined query (e.g., Happiest Country) for analysis. The main section displays data-loading status, options for selecting features and target variables, and the performance metrics of the selected model.

	Country_name	Regional_indicator	Ladder_score	upperwhisker	lowerwhisker	Log_GDP_pc
0	Finland	Western Europe	7.741	7.815	7.667	
1	Denmark	Western Europe	7.583	7.665	7.5	
2	Iceland	Western Europe	7.525	7.618	7.433	
3	Sweden	Western Europe	7.344	7.422	7.267	
4	Israel	Middle East and North Africa	7.341	7.405	7.277	
5	Netherlands	Western Europe	7.319	7.383	7.256	
6	Norway	Western Europe	7.302	7.389	7.215	
7	Luxembourg	Western Europe	7.122	7.213	7.031	
8	Switzerland	Western Europe	7.06	7.147	6.973	
9	Australia	North America and ANZ	7.057	7.141	6.973	

Figure 3.2- Show dataset

Figure 3.2 displays a portion of a **World Happiness Report Analysis** with key indicators for happiness across various countries. It includes data columns like **Country name**, **Regional indicator**, **Ladder score** (happiness score), and economic indicator **Log_GDP_per_capita**, among others. The table shows the top countries, mainly from **Western Europe**, ranked by happiness scores, with Finland at the top.

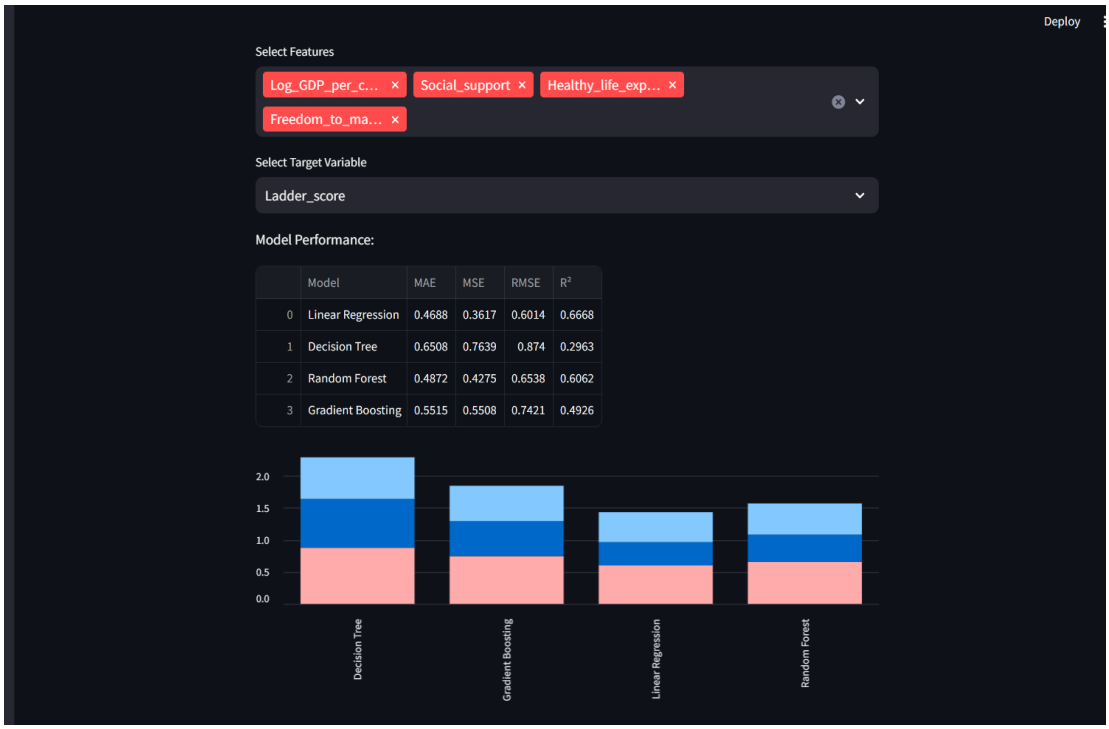

Figure 3.3-Model performance

Figure 3.3 shows a model performance comparison for predicting the Ladder score (happiness score) based on selected features like Log_GDP_per_capita, Social_support, and others. Four models are evaluated: Linear Regression, Decision Tree, Random Forest, and Gradient Boosting. Performance metrics such as MAE, MSE, RMSE, and R^2 are displayed, where Linear Regression performs best with the highest R^2 value (0.6668) and the lowest errors. A bar chart visualizes the comparison of model errors.

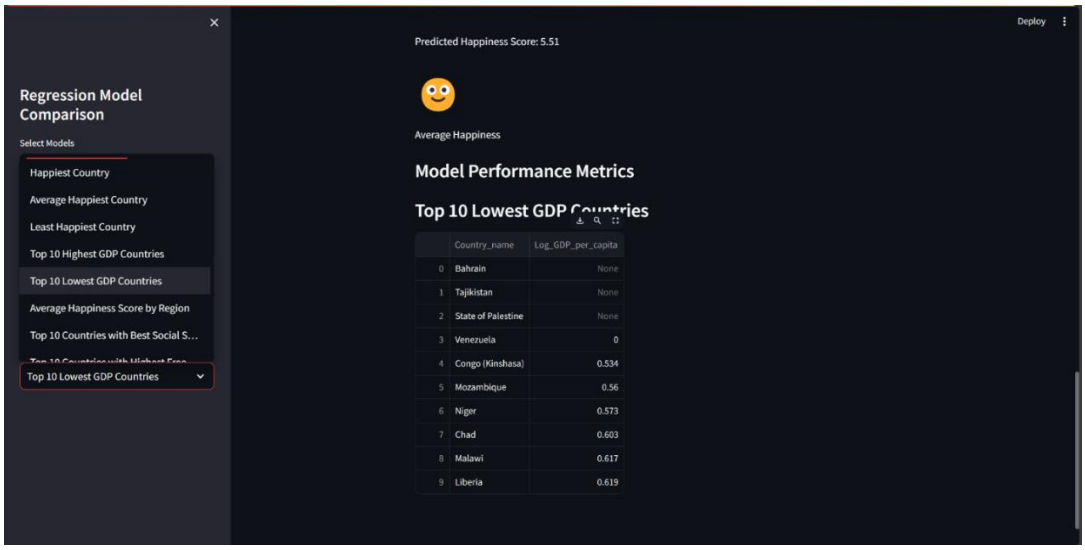

Figure 3.4-Query analysis

Figure 3.4 shows a dashboard comparing regression model outputs for happiness prediction. It displays the predicted happiness score of 5.51 and highlights the top 10 countries with the lowest GDP per capita, listing countries like Bahrain, Venezuela, and Liberia. The "Log_GDP_per_capita" column shows GDP values in logarithmic scale, with some countries having missing data. The side menu allows toggling between different metrics and categories, such as happiest countries and countries with the best social support

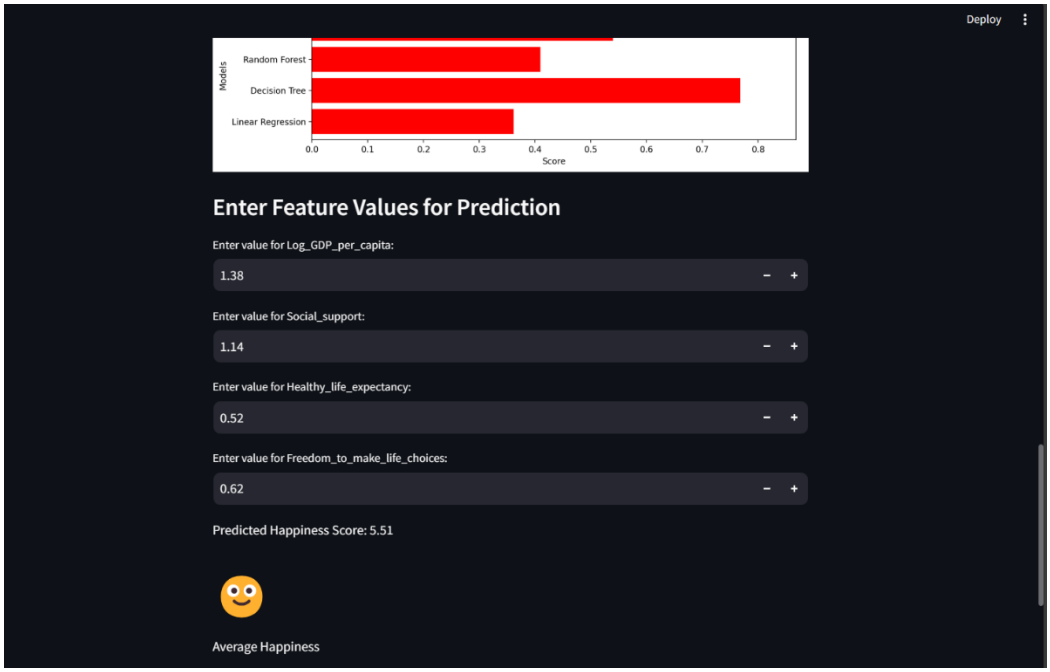

Figure 3.5-Feature value for prediction

Figure 3.5 shows a regression model comparison for predicting happiness scores. A bar graph compares Random Forest, Decision Tree, and Linear Regression models based on their performance scores. Below the graph, users can input feature values like "Log_GDP_per_capita","Social_support","Healthy_life_expectancy,"and"Freedom_to_make_life_choices" to predict happiness. The model outputs a predicted happiness score of 5.51 based on the entered features

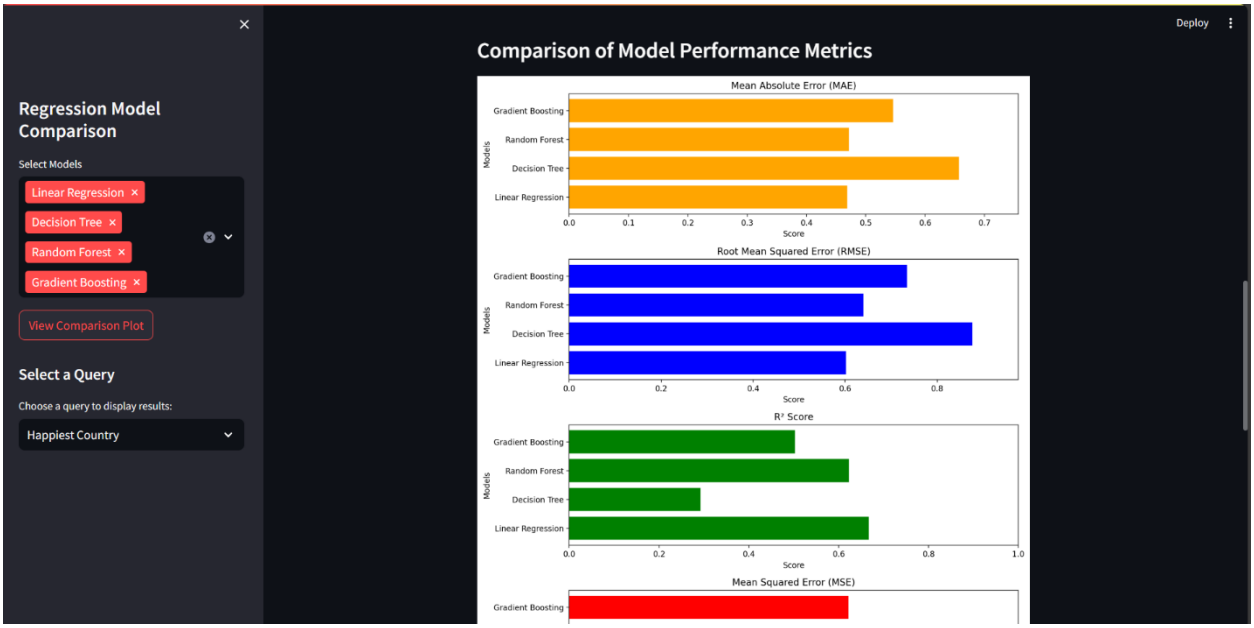

Figure 3.6-Comparison of models

Figure 3.6 shows a comparison of regression model performance metrics, evaluating four models: Linear Regression, Decision Tree, Random Forest, and Gradient Boosting. The metrics compared include Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), R^2 Score, and Mean Squared Error (MSE). Gradient Boosting consistently performs better across most metrics, exhibiting lower errors and higher R^2 values, indicating better predictive accuracy. The chart provides a visual overview of how each model performs on the given dataset, allowing for easy model comparison based on different evaluation criteria.

CHAPTER 4

CONCLUSION

The project successfully demonstrates an integrated web application that empowers users to analyze datasets through machine learning regression models and derive insights from the World Happiness Report. By utilizing the Streamlit framework, the application offers an intuitive interface that simplifies the process of model selection, training, and evaluation, making advanced analytics accessible to users without extensive programming knowledge.The implementation of various regression models—such as Linear Regression, Random Forest, and XGBoost—allows users to explore different modeling approaches and select the most suitable one for their specific dataset. This versatility ensures that users can address a wide range of predictive challenges. The application provides a robust evaluation framework that calculates essential performance metrics (MAE, MSE, RMSE, and R^2 score) for each model, enabling users to assess the effectiveness of their chosen regression technique and make informed decisions based on empirical results.Through SQL queries, users can extract meaningful insights from the World Happiness Report, highlighting relationships between happiness scores and various economic and social indicators. This functionality not only enhances understanding of global trends but also encourages data-driven discussions about factors influencing happiness.The Streamlit interface facilitates smooth user interaction, from uploading datasets to visualizing results, thereby reducing the technical barrier for users who may be new to data science and machine learning.

REFERENCE

- https://www.geeksforgeeks.org/world-happiness-index-2023/
- https://www.geeksforgeeks.org/z-score-for-outlier-detection-python/
- https://medium.com/@allysmatrix/data-analysis-and-eda-in-python-for-beginners-world-happiness-report-dataset-14d8840e3741
- https://medium.com/@eniola-emmanuel/world-happiness-data- analysis-using-python-53ce5cf03974
- https://www.kaggle.com/code/ismailsefa/world-happiness-data-analysis-and-visualization