INTERNSHIP

Submitted By

Rana Mitesh A.

(*226090307113*)

To

COMPUTER ENGINEERING DEPARTMENT

C U SHAH (GOVT.) POLYTECHNIC – SURENDRANAGAR

GUJARAT TECHNOLOGICAL UNIVERSITY – AHMEDABAD

JUNE-AUGUST – 2024

INDEX

Project Introduction

- Insurance charges prediction in machine learning is a crucial application of predictive analytics in the healthcare industry. The primary goal is to accurately forecast insurance costs, enabling insurance companies to offer personalized policies and optimize their business strategies. By leveraging machine learning algorithms, insurers can analyse various factors, such as demographics, medical history, and lifestyle, to predict the likelihood of certain health outcomes and associated costs.

- The insurance charges prediction model can be developed using various machine learning algorithms, including linear regression, decision trees, random forests, support vector machines, and neural networks. The choice of algorithm depends on the complexity of the data, the number of features, and the desired level of accuracy.

Project advantages and limitations

Advantages: -

I. Improved Accuracy:
The project's machine learning algorithms enable insurers to predict insurance charges with higher accuracy, reducing the risk of under or overcharging customers.

II. Personalization:
The project's customer segmentation and customized policies enable insurers to offer personalized policies that cater to individual needs and risk profiles.

III. Increased Transparency:
The project's transparent pricing models and model interpretability enable customers to understand how their premiums are calculated, increasing trust and transparency in the insurance industry.

IV. Cost Savings:
The project's accurate predictions enable insurers to optimize their business strategies, reducing costs and improving profitability.

V. Enhanced Customer Experience:
The project's personalized policies and transparent pricing models improve the overall customer experience, increasing customer satisfaction and loyalty.

VI. Competitive Advantage:
Insurers that adopt the project's insurance charges prediction model can gain a competitive advantage in the market, attracting more customers and increasing market share.

VII. Improved Healthcare Outcomes:
The project's accurate predictions enable insurers to identify high-risk customers and offer targeted interventions, improving healthcare outcomes and reducing healthcare costs

Limitations: -

I. Data Quality Issues:

The project's accuracy is dependent on the quality of the data used to train the machine learning models. Poor data quality can lead to inaccurate predictions.

II. Model Bias:

The project's machine learning models may be biased towards certain customer segments or demographics, leading to unfair treatment of certain groups.

III. Regulatory Compliance:

The project must ensure compliance with relevant regulations, such as HIPAA, which can be time-consuming and costly.

IV. Integration Challenges:

The project's API integration may require significant changes to an insurer's existing systems, which can be complex and time-consuming.

V. Customer Acceptance:

Some customers may be resistant to the use of machine learning algorithms and data analytics in insurance pricing, which can affect adoption rates.

VI. Model Maintenance:

The project's machine learning models require ongoing maintenance and updates to ensure they remain accurate and effective, which can be resource-intensive.

VII. Cybersecurity Risks:

The project's use of cloud-based infrastructure and data analytics increases the risk of cybersecurity breaches, which can compromise sensitive customer information.

Project application(s)

I. Health Insurance:

Predicting healthcare costs for individuals and groups, enabling insurers to offer personalized policies and optimize their business strategies.
Identifying high-risk customers and offering targeted interventions to improve healthcare outcomes and reduce healthcare costs.

II. Life Insurance:

Predicting life expectancy and mortality rates, enabling insurers to offer personalized policies and optimize their business strategies.
Identifying high-risk customers and offering targeted interventions to improve life expectancy and reduce mortality rates.

III. Auto Insurance:

Predicting accident risks and repair costs, enabling insurers to offer personalized policies and optimize their business strategies.
Identifying high-risk drivers and offering targeted interventions to improve road safety and reduce accident rates.

IV. Home Insurance:

Predicting property damage and repair costs, enabling insurers to offer personalized policies and optimize their business strategies.
Identifying high-risk properties and offering targeted interventions to improve property safety and reduce damage rates.

V. Government Agencies:

Predicting healthcare costs and identifying high-risk populations, enabling government agencies to optimize their healthcare programs and allocate resources more effectively.
Predicting insurance fraud and identifying high-risk claims, enabling government agencies to reduce fraud and improve the efficiency of their insurance programs.

VI. Research Institutions:

Predicting healthcare outcomes and identifying high-risk populations, enabling research institutions to develop more effective treatments and interventions.

Frontend and Back-end Technologies of project

Front-End Technologies: -

- ✓ **Matplotlib**:
  - o Specifically used in your code for plotting bar graphs, scatter plots, and pie charts.
- ✓ **Seaborn**:
  - A Python visualization library based on Matplotlib that provides a high-level interface for drawing attractive and informative statistical graphics.
  - Used for heatmaps and other types of plots, providing more advanced visualizations.
- ✓ **YData Profiling (formerly Pandas Profiling)**:
  - Generates a report for exploratory data analysis (EDA) with various visualizations and insights.
  - The Profile Report(df) generates an EDA report of the dataset.

Back-end Technologies: -

1. **Pandas**:
   - A data manipulation and analysis library used for handling and analysing data structures.
   - It is the core library used for reading the dataset (df = pd.read_csv('Mobile_Project.csv')), and performing operations like grouping, filtering, and summarizing data.
2. **NumPy**:
   - Used for array operations, reshaping data, and performing mathematical calculations.
3. **Scikit-Learn**:
   - A Python library for machine learning that includes tools for data pre-processing, model building, and evaluation.
   - In your code, it is used for encoding categorical variables (LabelEncoder, OneHotEncoder), splitting the dataset (train_test_split), scaling data (RobustScaler), and building a linear regression model.
4. **Linear Algebra Operations**:
   - Performed using NumPy for calculating coefficients and intercepts in linear regression manually, and for predicting values.

Hardware and software requirements

Hardware Requirements: -

1. **Processor:**
   - A multi-core processor (e.g., Intel Core i5 or higher) to handle data processing tasks efficiently.
2. **Memory (RAM):**
   - At least 8 GB of RAM is recommended for handling large datasets and running machine learning models smoothly. For more extensive datasets, 16 GB or more would be beneficial.
3. **Storage:**
   - A minimum of 256 GB SSD for faster data access and overall system performance. If you're working with very large datasets, consider at least 512 GB or an additional external storage solution.

Software Requirements: -

1. **Operating System:**
   - Windows, macOS, or Linux. Ensure that the operating system is up-to-date to support the necessary software and libraries.
2. **Python Environment:**

   o **Python 3.7 or higher:** Required for running the code.
3. **IDE/Code Editor:**

   o **Jupyter Notebook:** For interactive data exploration and testing code snippets.

4. **Python Libraries:**
   - **Data Manipulation & Analysis:**
     1. NumPy: For numerical computations.
     2. pandas: For data manipulation and analysis.
5. **Data Visualization:**
   - matplotlib: For basic plotting. o seaborn: For more advanced and aesthetically pleasing visualizations.
   - missing no: For visualizing missing data.
   - ydata_profiling: For generating detailed profiling reports of the dataset.

Functional and Non-Functional requirements

Functional Requirements: -

1. **Data Loading and Preprocessing:**
    - Load and inspect the diamond dataset. o Handle missing values, if any. o Encode categorical variables (like cut, colour, clarity).
    - Scale features for better model performance.
2. **Exploratory Data Analysis (EDA):**
    - Visualize the relationships between different features (e.g., carat, cut, price). o Generate heatmaps to understand correlations between numerical features.
    - Create bar plots, scatter plots, and pie charts for deeper insights into the dataset.
3. **Model Building:**
    - Implement Linear Regression using a custom approach to calculate coefficients and intercepts manually.
    - Train and test the model on the processed dataset. o Evaluate the model's performance using metrics like Mean Squared Error (MSE), Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared.
4. **Prediction:**
    - Predict diamond prices using the trained model.
    - Visualize actual vs. predicted prices to assess model accuracy.
5. **Feature Engineering:**
    - Select relevant features (carat, color, clarity, etc.) for model training.
    - Perform One-Hot Encoding for categorical variables.
6. **Model Evaluation:**
    - Use appropriate metrics to assess the model's performance on test data.
    - Visualize the difference between actual and predicted prices.

Non-Functional Requirements: -

1. **Performance:**
   - Ensure the model can process and make predictions efficiently, even with a large dataset. o Optimize code to reduce computation time, especially during model training and evaluation.
2. **Scalability:**
   - Design the system to handle increasing amounts of data without significant performance degradation.
   - Ensure that the model can be easily updated or retrained with new data.
3. **Usability:**
   - Ensure the code is well-documented and easy to understand for future users or developers.
   - Create visualizations that are clear and provide valuable insights into the data.
4. **Maintainability:**
   - Structure the code in a modular way, making it easy to update or replace parts of the system. o Use clear and consistent naming conventions and comments to make the codebase easy to maintain.

5. **Reliability:**
   - Implement checks and validation steps to ensure the model's predictions are accurate and consistent.
   - Handle any potential errors or exceptions gracefully.

List of Users and Use case Diagrams for each user

1. Data Scientist/ML Engineer

- **Load Data:** Load the diamond dataset.
- **Preprocess Data:** Handle missing values, encode categorical variables, and scale features.
- **Data Exploration and Visualization:** Generate descriptive statistics and visualizations to understand data distribution.
- **Train Model:** Train machine learning models like Linear Regression.
- **Evaluate Model:** Use metrics like MSE, MAE, RMSE, and R^2 to evaluate model performance.
- **Fine-Tune Model:** Adjust hyperparameters or preprocessing steps to improve model accuracy.

2. Business Analyst

- **Review Visualizations:** Analyse various plots like heatmaps, scatter plots, and bar plots for insights.
- **Interpret Results:** Understand the model's predictions and their implications for diamond pricing strategies.
- **Generate Reports:** Use the findings to create business reports or presentations.

3. End User (Insurance Agents)

Use Case: Advise Clients

- **Description:** Insurance agents use the project's predictions to advise clients on the best policies for their needs.
- **Actors:** Insurance Agents
- **Preconditions:** Insurance agents have access to the project's API and have received insurance charge predictions.
- **Triggers:** Clients request insurance advice.
- **Postconditions:** Insurance agents provide personalized advice to clients based on the project's predictions.

Data Dictionary

Column Name	Data Type	Description
Age	int	Policyholder's age in years
Gender	varchar	Policyholder's gender (Male/Female)
BMI	float	Policyholder's body mass index (BMI)
Smoker	Boolean	Binary variable indicating whether the policyholder smokes (Yes/No)

Sample coding

Step 1: - Import Libraries

→ import numpyas np
→ import pandas as pd
→ import matplotlib.pyplot as plt
→ import seaborn as sns
→ from sklearn.model_selection import train_test_split
→ from sklearn.preprocessing import LabelEncoder, StandardScaler
→ from sklearn.linear_model import LinearRegression
→ from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
→ import warnings
→ warnings.filterwarnings("ignore")

Step 2: - Load the Dataset

→ # Load the Mobiles dataset
→ df = pd.read_csv('Insurance_Project.csv')
→ print(df.head())
→ print(df.info())
→ print(df.describe())

Step 3: - Data Preprocessing

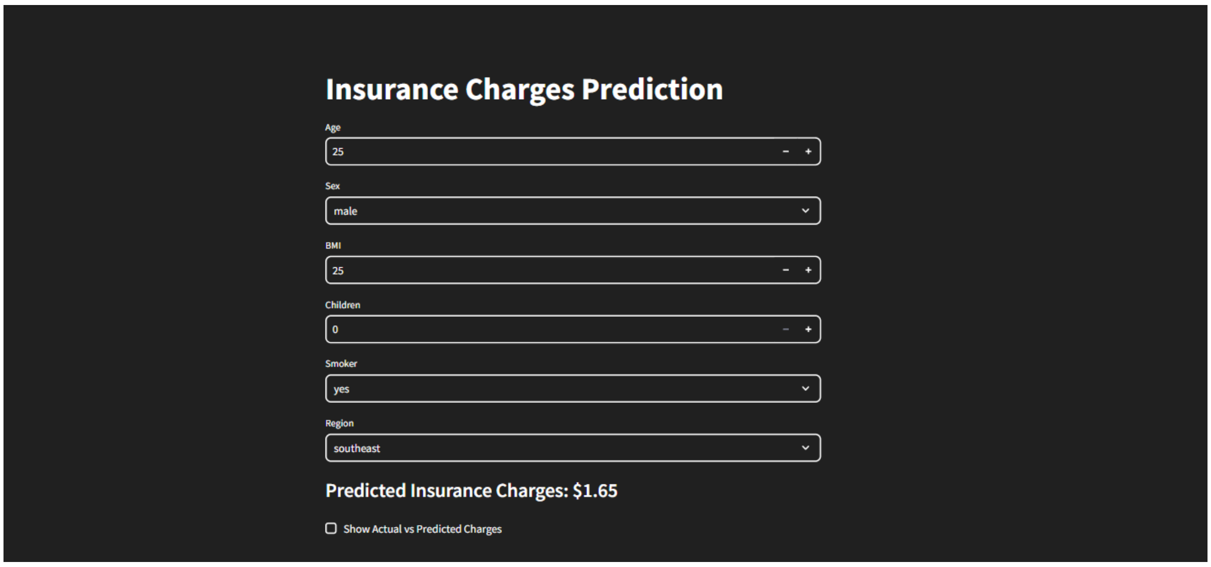
→ #Checking for missing values
→ print(df.isnull().sum())
→ # Encode categorical variables
→ label_enc = LabelEncoder()
→ df['cut'] = label_enc.fit_transform(df['cut'])
→ df['color'] = label_enc.fit_transform(df['color'])
→ df['clarity'] = label_enc.fit_transform(df['clarity'])
→ # Feature selection
→ X=dataset.drop('price_range',axis=1)
→ y=dataset['price_range']
→ # Split the dataset into training and testing sets
→ from sklearn.model_selection import train_test_split
→ X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

Step 4: - Creating And Training Linear Regression Model

→ # Initialize the Linear Regression model
→ from sklearn.linear_model import LinearRegression
→ lm = LinearRegression()
→ lm.fit(X_train,y_train)
→ lm.score(X_test,y_test)

Screen shot of various output pages

Without output Prediction Screen Shot: -



Insurance Charges Prediction

Age

25

Sex

male

BMI

25

Children

0

Smoker

yes

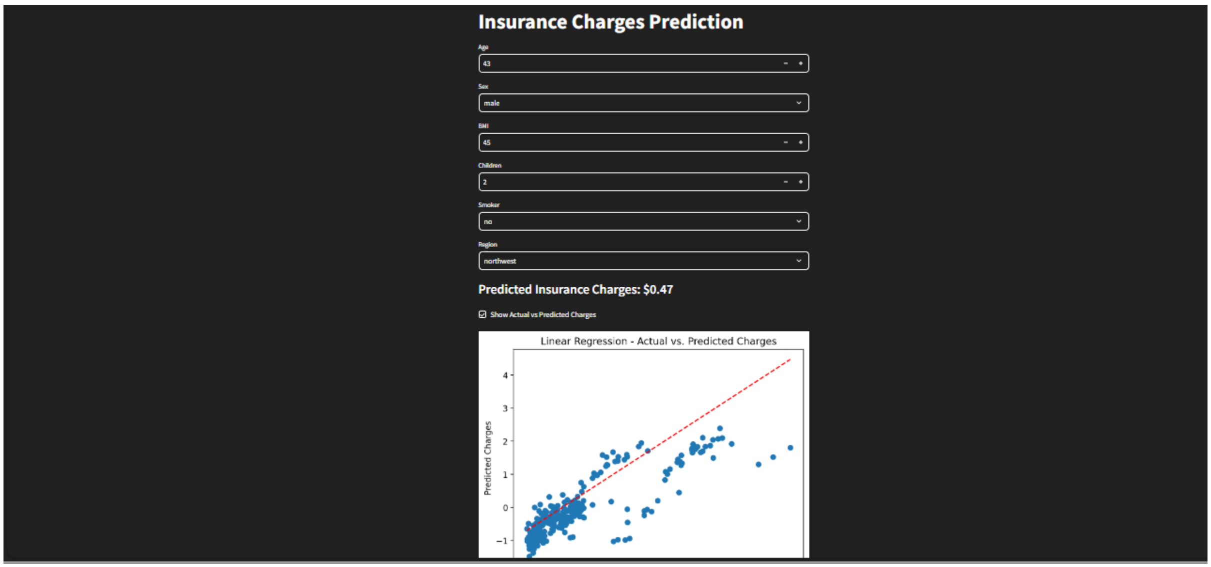
Region

southeast

Predicted Insurance Charges: $1.65

☐ Show Actual vs Predicted Charges

With output Prediction Screen Shot: -



Insurance Charges Prediction

Age

43

Sex

male

BMI

45

Children

2

Smoker

no

Region

northwest

Predicted Insurance Charges: $0.47

☑ Show Actual vs Predicted Charges

Testing strategies

1. Data Loading and Preprocessing: -

- **Loading the Dataset:** The dataset is loaded into a panda Data Frame.
- **Data Cleaning:** The code checks for null values and unique values in categorical columns like cut, clarity, and color. Missing data visualization is also performed using missingno.
- **Data Encoding:** Categorical columns are encoded using LabelEncoder and OneHotEncoder. The encoded features are then added back to the DataFrame, and the original categorical columns are dropped.
- **Feature Selection:** Features such as carat, color, clarity, depth, table, x, y, z, and the one- hot encoded cut features are selected as input features (X). The target variable is price.

2. Data Splitting: -

- **Train-Test Split:** The dataset is split into training and testing sets using an 80-20 split. The train_test_split function is used to create x_train, x_test, y_train, and y_test.

3. Data Scaling: -

- **Robust Scaling:** The RobustScaler is applied to scale the features and target variable.
- This scaling technique is less sensitive to outliers, making it suitable for this kind of data.

4. Model Building: -

Linear Regression (Manual Calculation):

- The code calculates the slope (coefficient) and intercept of the linear regression line manually using the training data.
- Predicted values (y and z) are calculated using the linear regression equation $y=mx+cy = mx + cy=mx+c$.

5. Model Evaluation: -

- **Performance Metrics:** o **Mean Squared Error (MSE):** Measures the average squared difference between actual and predicted values.
- **Mean Absolute Error (MAE):** Measures the average absolute difference between actual and predicted values.
- **Root Mean Squared Error (RMSE):** Provides the square root of MSE, making the error metric interpretable in the same units as the target variable.

- **Visualization:** The code plots a scatter plot comparing the actual vs. predicted prices to visualize the performance of the model.

6. Linear Regression Using Scikit-Learn: -

- **Model Training:** The LinearRegression model is trained using the scaled training data. o **Prediction:** Predictions (y_hat) are made on the test set. o **R-Squared Score:** The R^2 score is computed to evaluate the model's performance, indicating the proportion of variance in the target variable that is predictable from the input features.

Conclusion and Future work

Conclusion: -

1. **Data Exploration:**
    - The dataset contains 53,940 rows and 10 columns with a mix of numeric and categorical features.
    - summary statistics.
2. **Visualizations:**
    - **Carat vs Price:** A scatter plot indicates a strong positive correlation between carat weight and price.
    - **Cut vs Price:** A bar plot suggests variations in price based on the cut quality of the diamond.
    - **Clarity vs Price:** A pie chart shows the distribution of total prices across different clarity grades.
    - **Depth vs Price:** Scatter plot shows how depth relates to price, though the relationship might not be as strong.
3. **Data Preprocessing:**
    - Categorical variables ('cut', 'color', and 'clarity') were encoded using LabelEncoder and OneHotEncoder.
    - Feature scaling was performed using RobustScaler to handle outliers and scale the features appropriately.
4. **Modeling:**
    - **Mean Squared Error (MSE):** Provides an idea of the average squared difference between actual and predicted values.
    - **Mean Absolute Error (MAE):** Indicates the average absolute difference between actual and predicted values.
    - **Root Mean Squared Error (RMSE):** The square root of MSE, offering an error metric in the same units as the target variable.
    - The r2_score was used to assess how well the model explains the variance in the target variable.
5. **Results:**
    - The linear regression model provided a basic but effective approach to predicting diamond prices.
    - Visualizations and error metrics suggest that the model captures some of the relationships between features and price but may have room for improvement.

Future work: -

1. **Model Improvement:**
    - **Feature Engineering:** Create additional features or interactions between features that may improve model performance.
    - **Advanced Models:** Explore more complex models such as Ridge or Lasso regression, decision trees, random forests, or gradient boosting algorithms for potentially better results.
2. **Hyperparameter Tuning:**
    - Perform hyperparameter tuning for the models to find the best parameters that improve performance.
3. **Cross-Validation:**
    - Implement cross-validation to ensure the model's performance is robust and not overly fitted to the training data.
4. **Data Enrichment:**
    - Collect more data or include additional features that might impact diamond prices, such as market trends, brand, or diamond shape.
5. **Outlier Handling:**
    - Investigate and handle outliers in the data to ensure they do not disproportionately affect model performance.
6. **Feature Importance Analysis:**
    - Use techniques like feature importance from tree-based models to understand which features have the most impact on price prediction.
7. **Deployment:**
    - Develop a user-friendly interface for the model, potentially using Streamlit or other frameworks, to allow users to input diamond features and receive price predictions.
8. **Model Evaluation:**
    - Continuously evaluate the model's performance with new data and refine the model as needed based on real-world feedback and additional data.

References

- ✓ https://www.kaggle.com/
- ✓ https://www.kaggle.com/code/karnikakapoor/diamond-price- prediction
- ✓ https://scikit-learn.sourceforge.net/dev/index.html
- ✓ https://www.w3schools.com/python/pandas/default.asp
- ✓ **https://chatgpt.com/**
- ✓ **https://github.com/**