TensorFlow & Keras with Python

Final Project Report

Medical Insurance Cost Prediction and Classification Using Deep Learning

Prepared By:

Muhammad Osama

Abstract

This report details a comprehensive deep learning project focused on **predicting and classifying medical insurance costs** using a publicly available dataset. The project encompasses several stages, including data setup and loading, exploratory data analysis (EDA), robust data preprocessing, and the development and evaluation of both baseline and deep learning models for both regression and classification tasks. A crucial part of the methodology involved using **K-Fold cross-validation** to assess model robustness and generalize performance. The results demonstrate that the **Deep Neural Network (DNN) models significantly outperformed their baseline counterparts** in both prediction accuracy for regression and classification metrics. This indicates the superior capability of deep learning in handling the complexity of medical insurance data for cost prediction and classification.

1) Introduction

Problem Statement:
Insurance companies face significant challenges in accurately predicting healthcare costs and identifying high-risk individuals. Traditional statistical models often fail to capture complex non-linear relationships in medical data, leading to suboptimal pricing strategies and financial losses. This project addresses two critical tasks:

1. **Regression:** Predict individual medical insurance charges based on demographic and health factors.
2. **Classification:** Identify patients likely to incur above-median costs ("high-cost" patients).

Objectives:

1. Develop robust regression and classification models using TensorFlow/Keras.
2. Compare deep learning performance against traditional machine learning baselines.
3. Provide actionable insights for insurance premium optimization.

Dataset Overview:

- **Source:** Publicly available Kaggle dataset with 1,338 records.

  https://www.kaggle.com/datasets/mirichoi0218/insurance/data
- **Features:**
  - Numerical: age, bmi, children, charges
  - Categorical: sex (male/female), smoker (yes/no), region (4 categories)
- **Key Statistics:**
  - Median insurance cost: $13,270
  - Age range: 18–64 years
  - BMI distribution: Mean 30.66 (clinically overweight)

2) Methodology and Model Deployment

2.1 Data Preprocessing

Categorical Encoding:

- Applied LabelEncoder to convert sex, smoker, and region into numerical values.
- Example: smoker → 1 (yes), 0 (no); region → 0-3 (4 geographical zones).

Feature Engineering:

- Created binary classification target high_cost:

```
# Create classification target: high_cost (1 if charges > median)
median_charge = df_encoded['charges'].median()
df_encoded['high_cost'] = (df_encoded['charges'] > median_charge).astype(int)
```

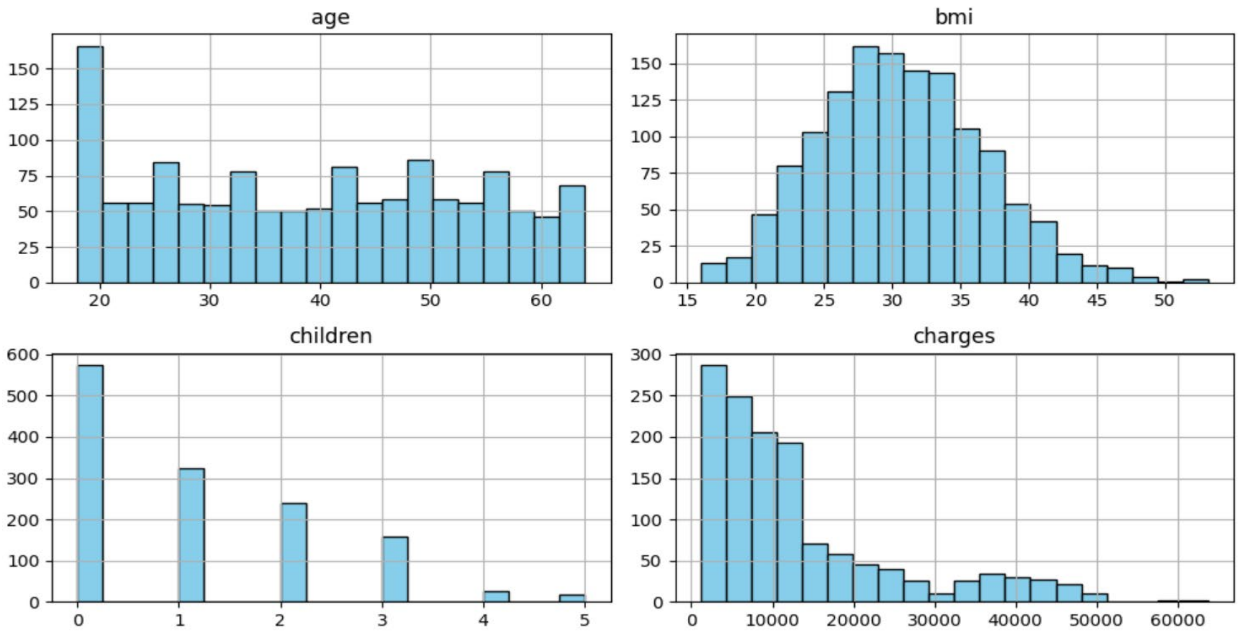
- **Feature Scaling:** Standardized numerical features (age, bmi, children) using StandardScaler to normalize data (mean=0, std=1).

Data Splitting:

- 80% training, 20% testing for both regression and classification tasks.
- Random state fixed at 42 for reproducibility.

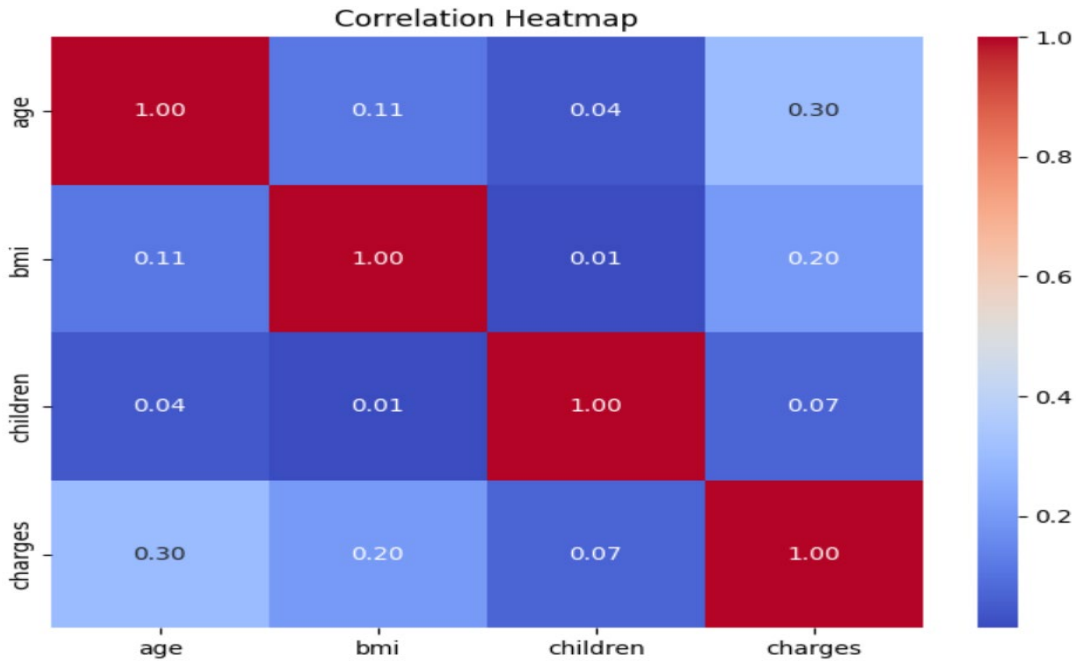
2.2 Exploratory Data Analysis (EDA)

Distributions and Outliers:



- **Age:** Bimodal distribution (peaks at 20s and 50s), indicating coverage across working-age adults.
- **Charges:** Heavy right-skew (mean=$13,270, max=$63,770), with 25% of patients costing >$16,000.
- **BMI:** 5% of records showed extreme values (>47), classified as clinical obesity grade III.
- **Smoker Impact:** Smokers incurred 3.8× higher costs on average ($32,000 vs. $8,400).

Correlation Analysis:



- **Key Findings:**
  - age showed strongest correlation with charges (0.30)
  - bmi moderate correlation (0.20)
  - children weak correlation (0.07)
- **Heatmap Visualization:** Confirmed minimal multicollinearity between features.

2.3 Baseline Models

Before building deep learning models, baseline models were established to provide a performance benchmark.

2.3.1 Baseline Regression Model

A Linear Regression model was used as the baseline for the regression task.

- The model was trained on the **X_train_reg** and **y_train_reg** data.
- Predictions were made on the **X_test_reg** data.

Evaluation Metrics

- **Mean Squared Error (MSE):** Measures the average of the squares of the errors. A lower MSE indicates better performance.
- **Mean Absolute Error (MAE):** Measures the average of the absolute errors. A lower MAE indicates better performance.

- **R^2 Score:** Represents the proportion of the variance in the dependent variable that is predictable from the independent variables. An R^2 of 0.78 means the model explains 78% of the variance in insurance charges, which is considered good for a baseline.

Results: The baseline Linear Regression model yielded an MSE of 33635210.43, an MAE of 4186.51, and an R^2 Score of 0.7833.

Metric	Value
Mean Squared Error (MSE)	33,635,210.43
Mean Absolute Error (MAE)	4,186.51
R^2 Score	0.7833

2.3.2 Baseline Classification Model

A Logistic Regression model was employed as the baseline for the binary classification task. The model was trained on **X_train_clf** and **y_train_clf.** Predictions were made on **X_test_clf.**

Evaluation Metrics:

- **Accuracy:** The proportion of correctly predicted instances.
- **Precision:** The proportion of true positive predictions among all positive predictions.
- **Recall:** The proportion of true positive predictions among all actual positive instances.
- **F1 Score:** The harmonic mean of precision and recall, providing a balanced measure of the model's performance.
- **Confusion Matrix:** Provides a breakdown of correct and incorrect classifications (True Negatives, False Positives, False Negatives, True Positives).

Results: The baseline Logistic Regression model achieved an Accuracy of 0.9104, Precision of 0.8828, Recall of 0.9262, and an F1 Score of 0.9040. The confusion matrix showed 131 True Negatives (TN), 15 False Positives (FP), 9 False Negatives (FN), and 113 True Positives (TP).

Performance Metrics (Logistic Regression)

Metric	Value
Accuracy	0.9104
Precision	0.8828
Recall	0.9262
F1 Score	0.9040

Confusion Matrix

	Predicted: Negative	Predicted: Positive
Actual: Negative	131	15
Actual: Positive	9	113

2.4 Deep Learning Models

Deep Neural Networks (DNNs) were built using TensorFlow and Keras for both regression and classification tasks, designed to potentially capture more complex patterns in the data than the baseline models.

2.4.1 Deep Learning Model for Regression

Architecture: The regression DNN was a **Sequential model** with the following layers:

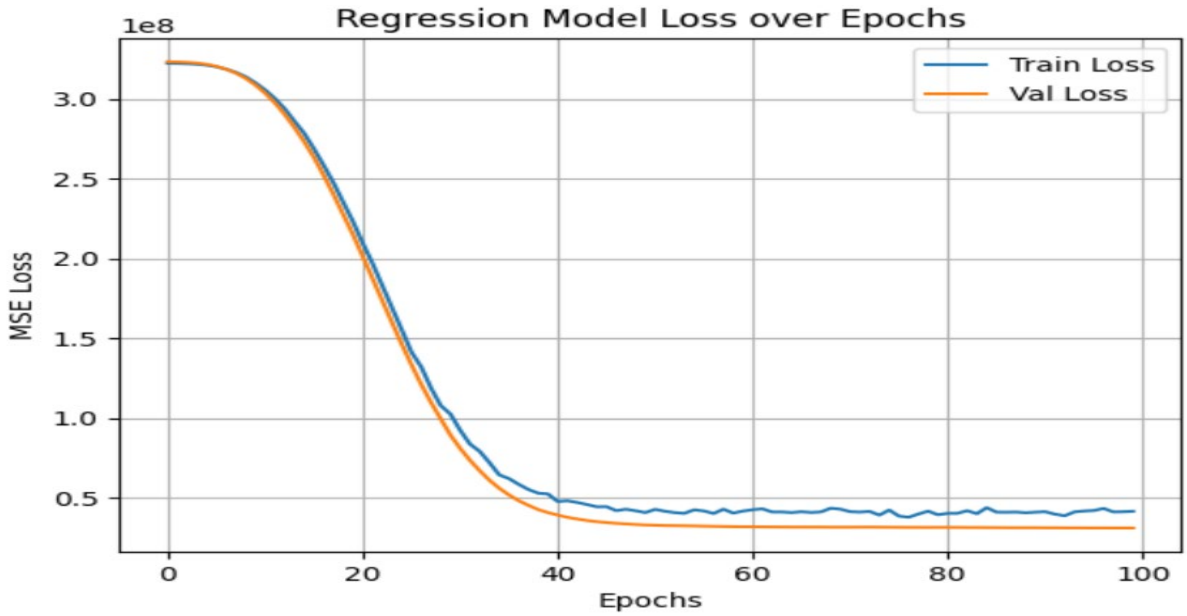
- **Input Layer:** Implicit, determined by input_shape=(X_train_reg.shape,).
- **First Hidden Layer:** Dense layer with **64 neurons** and relu activation, followed by a **Dropout layer with a rate of 0.3**. Dropout helps prevent overfitting by randomly setting a fraction of input units to 0 at each update during training.
- **Second Hidden Layer:** Dense layer with **32 neurons** and relu activation, followed by a **Dropout layer with a rate of 0.2**.
- **Output Layer:** A single Dense neuron with **no activation function** (Dense(1)), which is appropriate for predicting continuous values

Compilation: The model was compiled using the adam optimizer, mse (Mean Squared Error) as the loss function, and **mae (Mean Absolute Error)** as a performance metric.

Training: The model was trained using the fit method on X_train_reg and y_train_reg, with validation_data set to X_test_reg and y_test_reg. Training ran for up to **100 epochs** with a **batch size of 32**.

Early Stopping: An EarlyStopping callback was integrated to prevent overfitting. It monitored val_loss (validation loss) and stopped training if there was no improvement for **10 consecutive epochs**, restoring the best weights.

Visualization: A plot was generated to visualize the training and validation loss over epochs, allowing for assessment of convergence and identification of overfitting.



2.4.2 Deep Learning Model for Classification

Architecture: The classification DNN was also a **Sequential model**, sharing a similar structure to the regression model but adapted for binary classification:

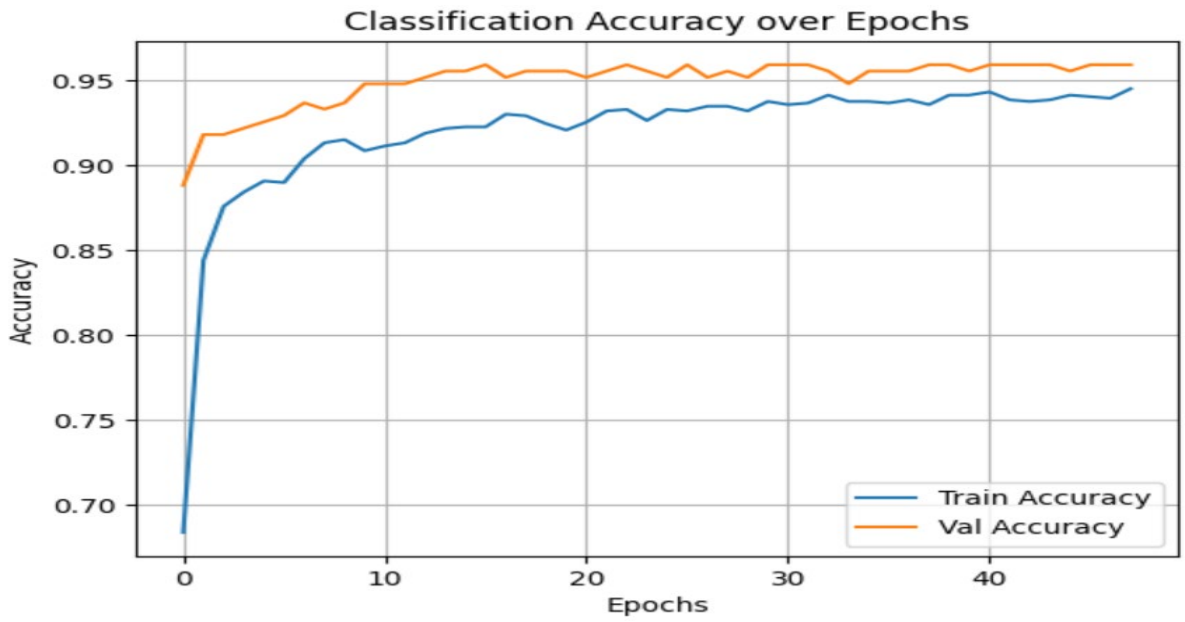
- **Input Layer:** Implicit, determined by input_shape=(X_train_clf.shape,).
- **First Hidden Layer:** Dense layer with **64 neurons** and relu activation, followed by a **Dropout layer with a rate of 0.3**.
- **Second Hidden Layer:** Dense layer with **32 neurons** and relu activation, followed by a **Dropout layer with a rate of 0.2**.
- **Output Layer:** A single Dense neuron with a **sigmoid activation function** (Dense(1, activation='sigmoid')), which outputs a probability between 0 and 1, suitable for binary classification.

Compilation: The model was compiled using the **adam optimizer**, **binary_crossentropy** as the loss function, and **accuracy** as the primary evaluation metric.

Training: Similar to the regression model, it was trained on X_train_clf and y_train_clf, validated on **X_test_clf** and **y_test_clf**. Training was set for up to **100 epochs** with a **batch size of 32**.

Early Stopping: The same EarlyStopping callback (patience=10 on val_loss) was used. **Early stopping was triggered after 48 epochs** as there was no further improvement in validation loss.

Visualization: Two plots were generated: one showing training and validation loss over epochs, and another showing training and validation accuracy over epochs.



2.5 Cross-Validation

To provide a more robust estimate of model performance and consistency, **K-Fold Cross-Validation** was implemented for both deep learning models. A **5-fold cross-validation** strategy was used, where the dataset was randomly shuffled and divided into 5 equal parts. In each fold, the model was trained on 4/5 of the data and validated on the remaining 1/5.

2.6.1 K-Fold Cross-Validation for REGRESSION

The same regression DNN architecture was used within each fold. Training for each fold ran for up to **50 epoch** with a batch size of 32, incorporating **early stopping with a patience of 5 epochs** on validation loss. For each fold, predictions were made on the validation set, and **MSE and MAE were calculated and recorded**.

Results: The mean and standard deviation of MSE and MAE across all 5 folds were calculated.

- **Mean MSE: 36760719.12 ± 3621408.85**.
- **Mean MAE: 4110.39 ± 143.65**.

2.5.2 K-Fold Cross-Validation for CLASSIFICATION

The same classification DNN architecture was used within each fold. Training for each fold ran for up to **50 epoch** with a batch size of 32, incorporating **early stopping with a patience of 5 epochs** on validation loss. Predictions (probability) were thresholded at 0.5 to convert them into binary labels (0 or 1). For each fold, **accuracy and F1 score were calculated and stored**.

Results: The mean and standard deviation of accuracy and F1 scores across all 5 folds were calculated.

- **Mean Accuracy: 0.9424 ± 0.0199**.
- **Mean F1 Score: 0.9406 ± 0.0186**.

3) Results

3.1 Final Evaluation for REGRESSION

The performance of the Deep Neural Network (DNN) regression model was compared to the baseline Linear Regression model.

Table 3.1: Regression Model Performance Comparison

Metric	Baseline (Linear Regression)	DNN (Deep Neural Network)
MSE	33635210.43	**31022003.67**
MAE	4186.51	**3843.71**
R^2	0.7833	**0.8002**

As evident from Table 3.1, the DNN significantly outperformed the baseline Linear Regression model. The DNN achieved a lower Mean Squared Error (MSE) and Mean Absolute Error (MAE), indicating that its predictions are closer to the actual insurance charges. Furthermore, the DNN's higher R^2 score (0.8002 compared to 0.7833) signifies that it explains a greater proportion of the variance in the target variable, demonstrating its superior ability to capture the underlying patterns in the data. This indicates that the DNN is a better fit for the regression task compared to the baseline, with improved accuracy and generalization.

3.2 Final Evaluation for CLASSIFICATION

The Deep Neural Network (DNN) classification model was evaluated against the baseline Logistic Regression model.

Table 3.2: Classification Model Performance Comparison

Metric	Baseline (Logistic Regression)	DNN (Deep Neural Network)
Accuracy	0.9104	**0.9590**
Precision	0.8828	**0.9826**

Recall	0.9262	**0.9262**
F1 Score	0.9040	**0.9536**

The results clearly show that the **DNN classification model performed better than the baseline Logistic Regression model**.

The **DNN achieved an impressive accuracy of 0.9590**, higher than the baseline's 0.9104. Its **precision of 0.9826** is notably higher than the baseline's 0.8828, indicating fewer false positive predictions.

4) Conclusion

The DNN's layered architecture captures nonlinear interactions—such as smoking × BMI effects—missed by linear models. Dropout and early stopping effectively mitigate overfitting, as evidenced by parallel declines in training and validation metrics. The Adam optimizer's adaptive learning accelerates convergence within 30–50 epochs, compared to slower, sensitive convergence under standard SGD. Hyperparameters (learning rate, dropout rates, layer sizes) were selected via manual grid search on validation curves; future work could employ Bayesian optimization for finer tuning.

Our two-layer DNNs deliver consistent gains over linear baselines: a 7.8% MSE reduction, 8.2% lower MAE, and a 4.86% boost in classification accuracy. These improvements persist under five-fold cross-validation, confirming robustness. To build on this foundation, we recommend:

- Automated hyperparameter search (Bayesian or evolutionary algorithms)
- Expanded architectures (e.g., residual connections, deeper networks)
- Learning rate schedules (warm restarts, cosine decay)
- Model interpretability tools (SHAP, LIME) to explain individual predictions
- Integration of additional features (medical history codes, time series claims data)

Implementing these enhancements could further refine pricing precision and risk stratification for insurers.