Medical Insurance Cost Prediction

Presented By: Muhammad Osama

Introduction

- **Objective:**

Predict insurance costs and classify high-cost patients.

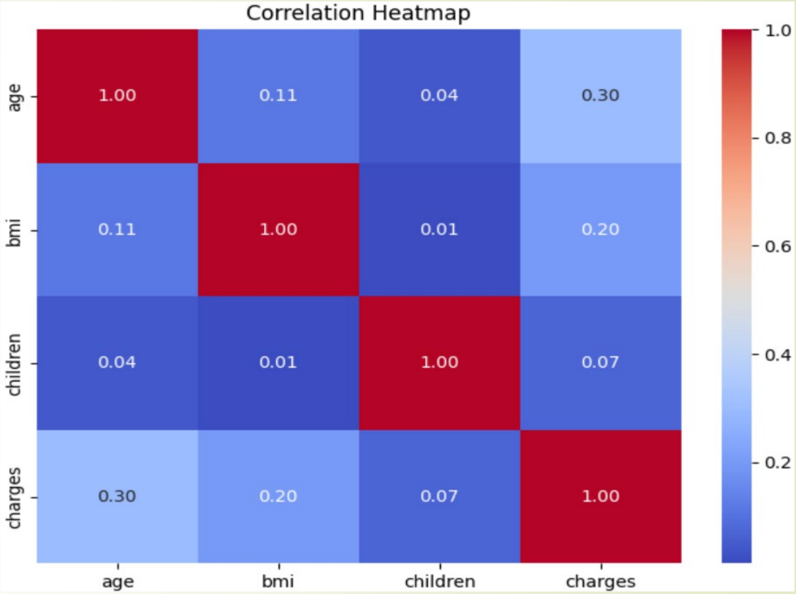
- **Dataset:**
  - Source: Kaggle
  - Size: 1338 records, 7 features.
- **Features**
  - Numerical: Age, BMI, Children, Charges
  - Categorical: Sex, Smoker, Region
- **Importance:**

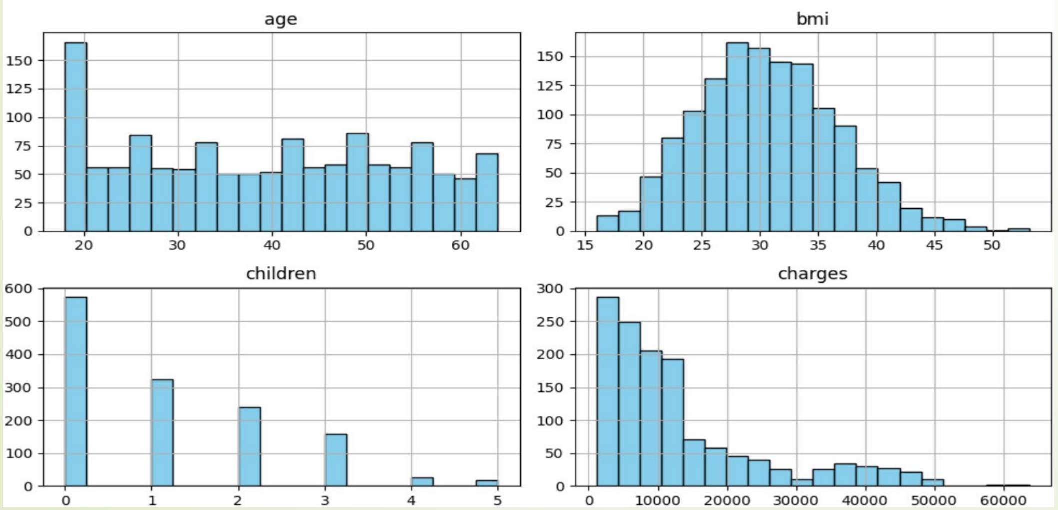
Insurance pricing struggles to reflect individual risk accurately.

Need for robust models to predict medical costs and identify high-risk individuals..

Exploratory Data Analysis (EDA)

- **Correlation Insights**
  - Age & Charges: Strongest correlation (0.30)
  - BMI: Moderate impact (0.20)
- **Distributions & Outliers:**
  - Age shows bimodal peaks (20s & 50s).
  - Charges are right-skewed (max = $63,770)
  - Smokers pay ~ 3.8x more on average





Data Preprocessing

- **Importing Libraries:** Importing all necessary libraries
- **Missing Values:** Check for missing values
- **Label Encoding:** Label Encoded all categorical features
- **Feature Scaling :** StandardScaler normalized input features
- **New Target:** high_cost = charges > median (binary classification)
- **Train Test Split:** 80% training, 20% testing.

```python
# Check missing values
print("Missing values:\n", df.isnull().sum())

# Encode categorical features
df_encoded = df.copy()
label_encoders = {}

for col in ['sex', 'smoker', 'region']:
    le = LabelEncoder()
    df_encoded[col] = le.fit_transform(df_encoded[col])
    label_encoders[col] = le

# 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)

# Features for both models
features = ['age', 'sex', 'bmi', 'children', 'smoker', 'region']

# Scale features
scaler = StandardScaler()
df_encoded[features] = scaler.fit_transform(df_encoded[features])

# Define regression and classification targets
X = df_encoded[features]
y_reg = df_encoded['charges']
y_clf = df_encoded['high_cost']

# Split data
X_train_reg, X_test_reg, y_train_reg, y_test_reg = train_test_split(X, y_reg, test_size=0.2, random_state=42)
X_train_clf, X_test_clf, y_train_clf, y_test_clf = train_test_split(X, y_clf, test_size=0.2, random_state=42)
```

Baseline Models

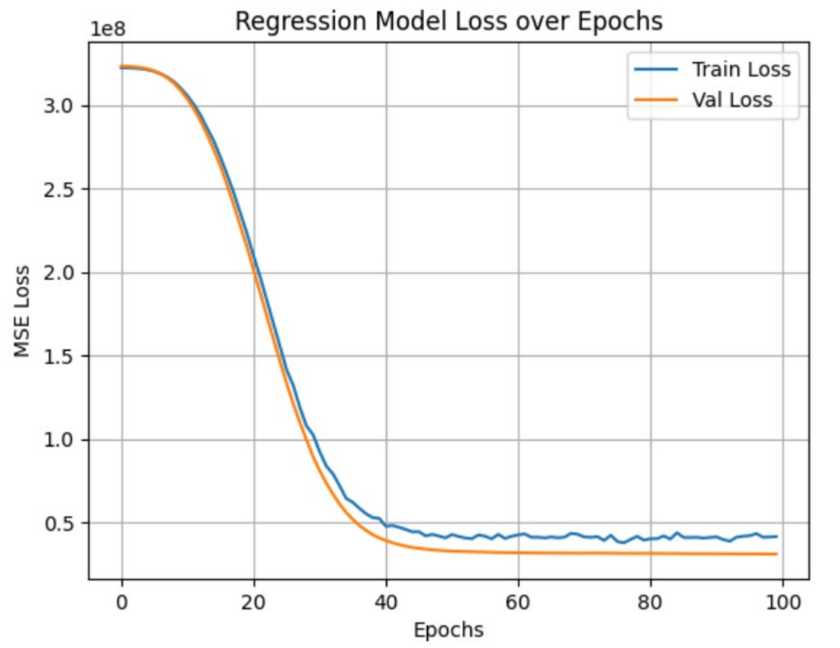
- **Linear Regression: For Cost Prediction**
  - **R^2 Score:** 0.783
  - **MSE:** 33635210
  - **MMAE:** 4186
- **Logistic Regression: For High-Cost classification**
  - **Accuracy:** 91%
  - **Precision:** 88%
  - **F1 Score:** 90%

📊 **Confusion Matrix Summary**

Metric	Value	Description
True Positives (TP)	131	Correctly predicted high-cost patients
False Positives (FP)	15	Predicted high-cost, but actually low-cost
False Negatives (FN)	9	Predicted low-cost, but actually high-cost
True Negatives (TN)	113	Correctly predicted low-cost patients

Deep Learning Models

- **Regression Model (DNN)**
  - **Architecture:** [64 → Dropout → 32 → Dropout → Output]
  - **Activation Function:** Relu
  - **Optimizer:** Adam
  - **Loss:** MSE
  - **Early Stopping**
  - **Performance:**
    - R^2 Score: 0.8002
    - MSE: 31022003.67
    - MAE: 3,843




```
reg_model = Sequential([
    Dense(64, activation='relu', input_shape=(X_train_reg.shape[1],)),
    Dropout(0.3),
    Dense(32, activation='relu'),
    Dropout(0.2),
    Dense(1)  # Output layer with 1 node for regression
])

reg_model.compile(optimizer='adam', loss='mse', metrics=['mae'])

# Callback to stop early if val_loss doesn't improve
early_stop = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)
```

Deep Learning Models

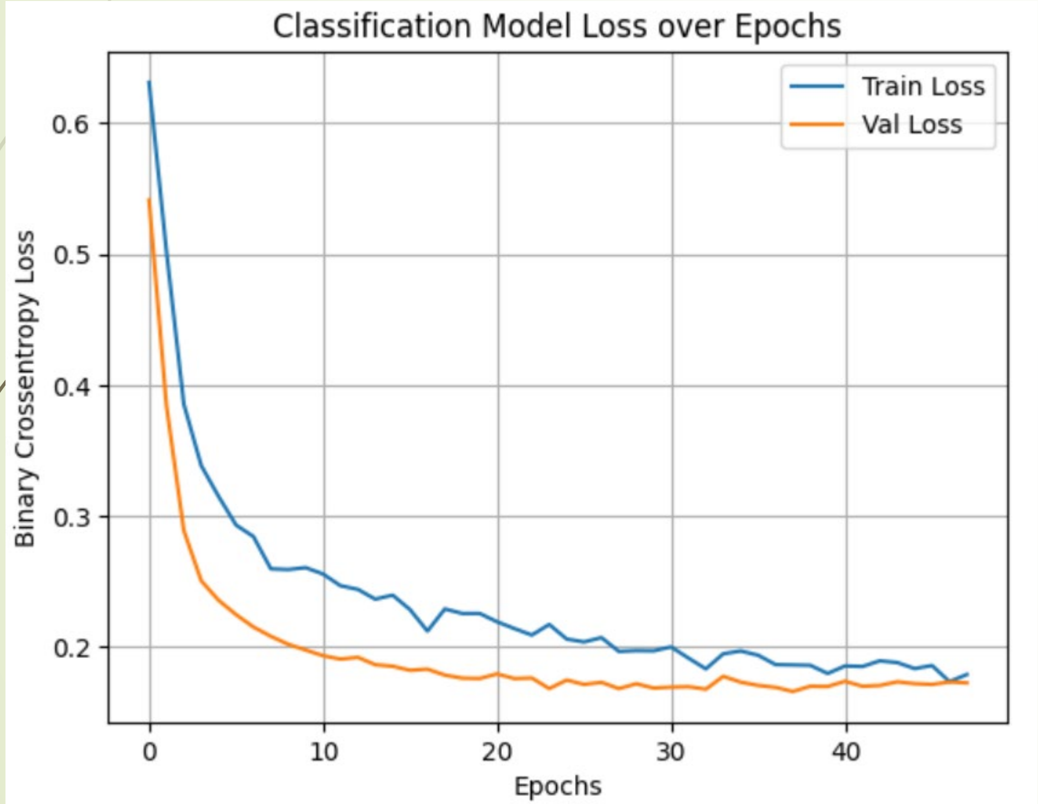
- **Classification Model (DNN)**
  - **Architecture:** [64 → Dropout → 32 → Dropout → Sigmoid]
  - **Activation Function:** Relu & Sigmoid
  - **Optimizer:** Adam
  - **Loss:** Binary Cross Entropy
  - **Early Stopping**
  - **Performance:**
    - Accuracy: 95.9 %
    - Precision: 98.3 %
    - F1 Score: 95.4 %

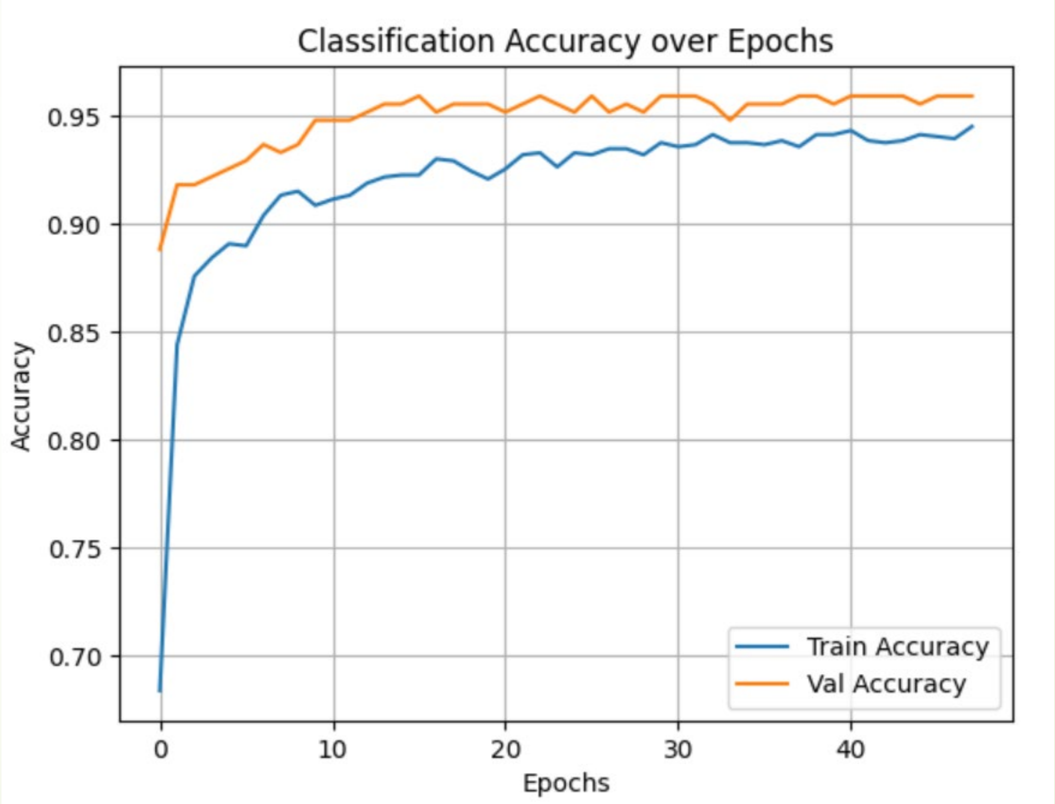
```python
# Build model
clf_model = Sequential([
    Dense(64, activation='relu', input_shape=(X_train_clf.shape[1],)),
    Dropout(0.3),
    Dense(32, activation='relu'),
    Dropout(0.2),
    Dense(1, activation='sigmoid')  # Sigmoid for binary classification
])

clf_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

Deep Learning Models

- **Classification Model (DNN) Loss & Accuracy Over Epoch**





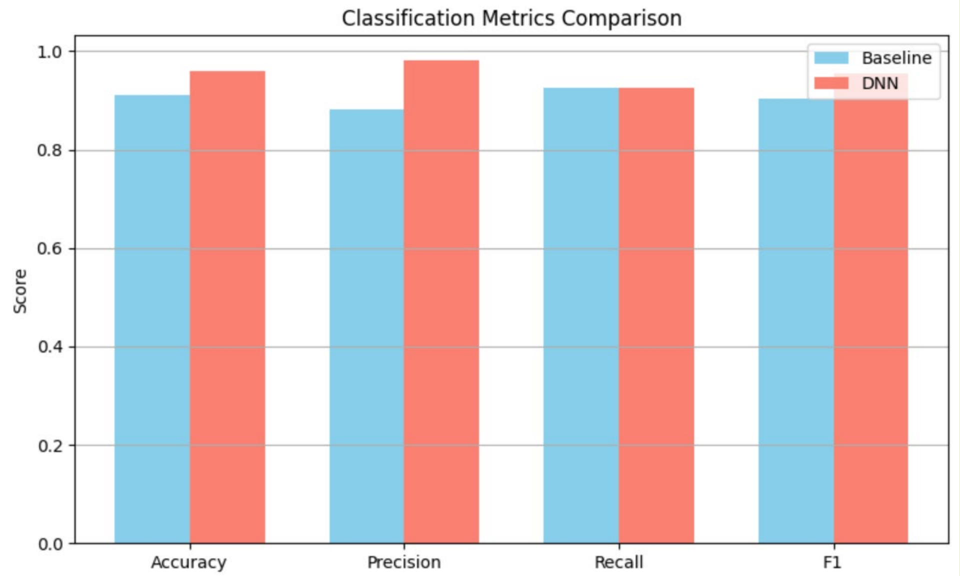
K-Fold Cross-Validation

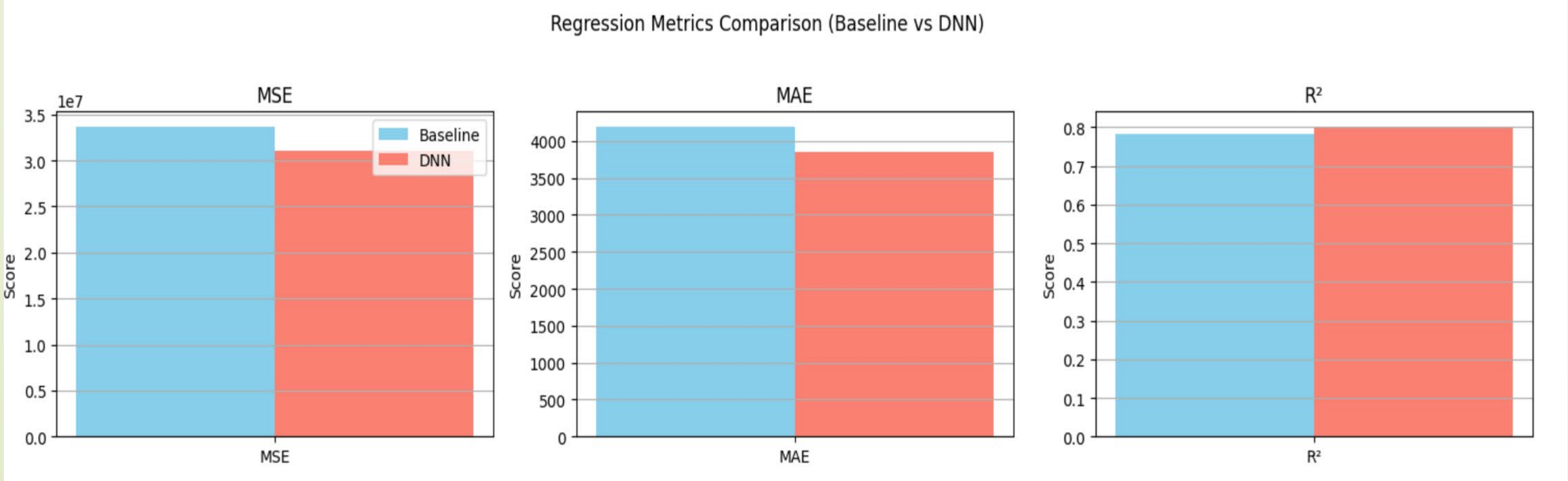
- A powerful method to test model stability and generalization
- The dataset is split into K equal parts (folds)
- The model trains on (K–1) folds and tests on the remaining one
- This cycle repeats K times, each fold playing the test set once
- Final performance is the average across all runs—reduces single-split
- **Results:**
- Classification DNN (5-Fold)
  - Mean Accuracy: 94.2% ± 2%
  - Mean F1: 94.0% ± 1.9%
- Regression DNN (5-Fold)
  - Mean MSE: 36.7M ± 3.6M
  - Mean MAE: 4,110 ± 144

Comparative Performance

Metric	Baseline (Regression)	DNN (Regression)
MSE	33.6M	31.0M
MAE	$4,186	$3,843
R^2 Score	0.783	0.800

Metric	Baseline (Classification)	DNN (Classification)
Accuracy	91%	95.9%
Precision	88.3%	98.3%
F1 Score	90.4%	95.4%





Conclusion

- **Deep Neural Networks outperform traditional models.**
- **DNNs capture non-linear patterns like smoker × BMI interactions.**
- **Recommendations:**
  - Use Bayesian optimization for hyperparameters
  - Integrate SHAP/LIME for model interpretability
  - Explore deeper or residual networks for further improvements

Thank you