MEDICAL INSURANCE COST PREDICTION

AN EXPERIMENT WITH SUPERVISED MODELS VS. DEEP LEARNING IN THE HEALTHCARE INDUSTRY

- Predicting Health Insurance Costs based on individual information and health risks.
- Compare traditional Machine Learning approaches and Deep Learning to discover advantages and disadvantages.

ABOUT THE PROJECT

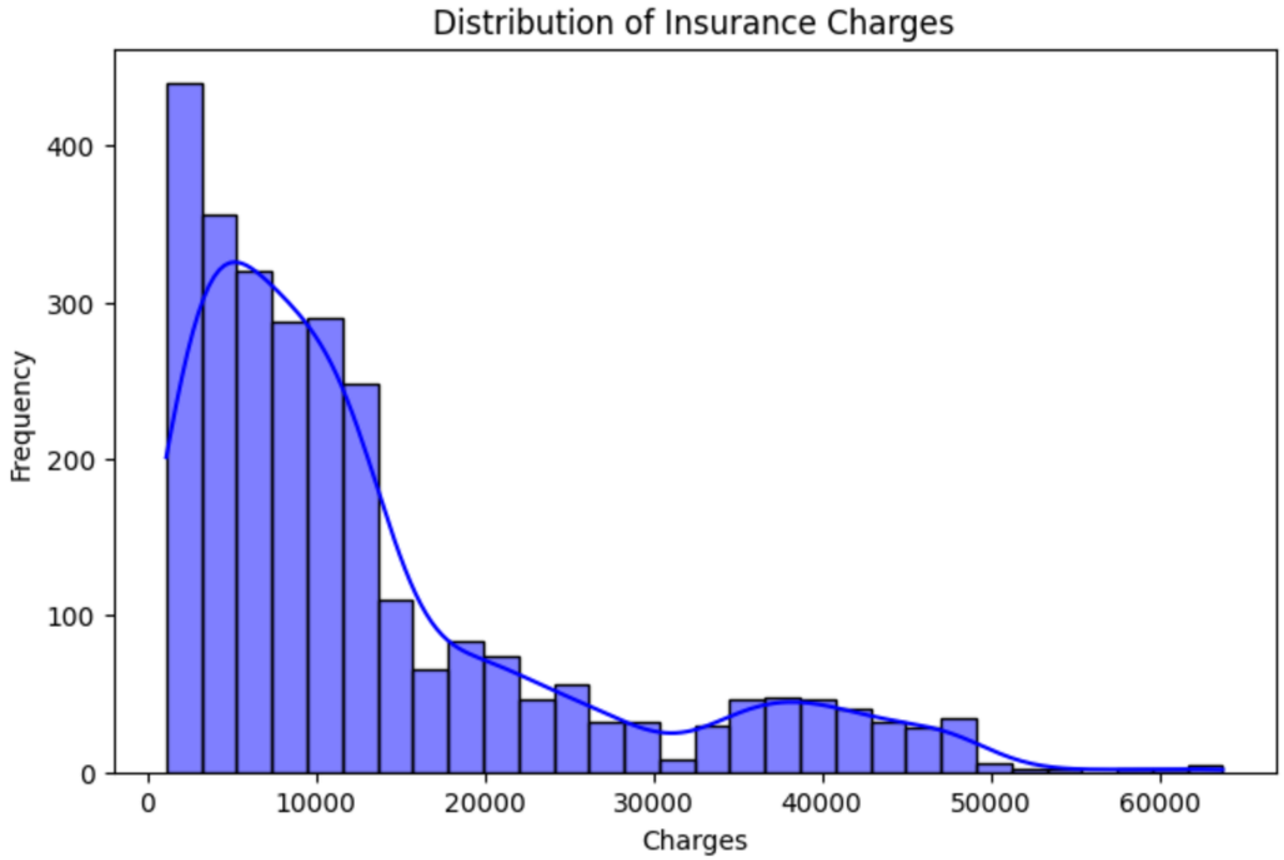
ABOUT THE DATA

- Health Insurance Cost dataset from Kaggle
- 2772 Observations, 7 Columns
- Pre-Cleaned (Null Values)

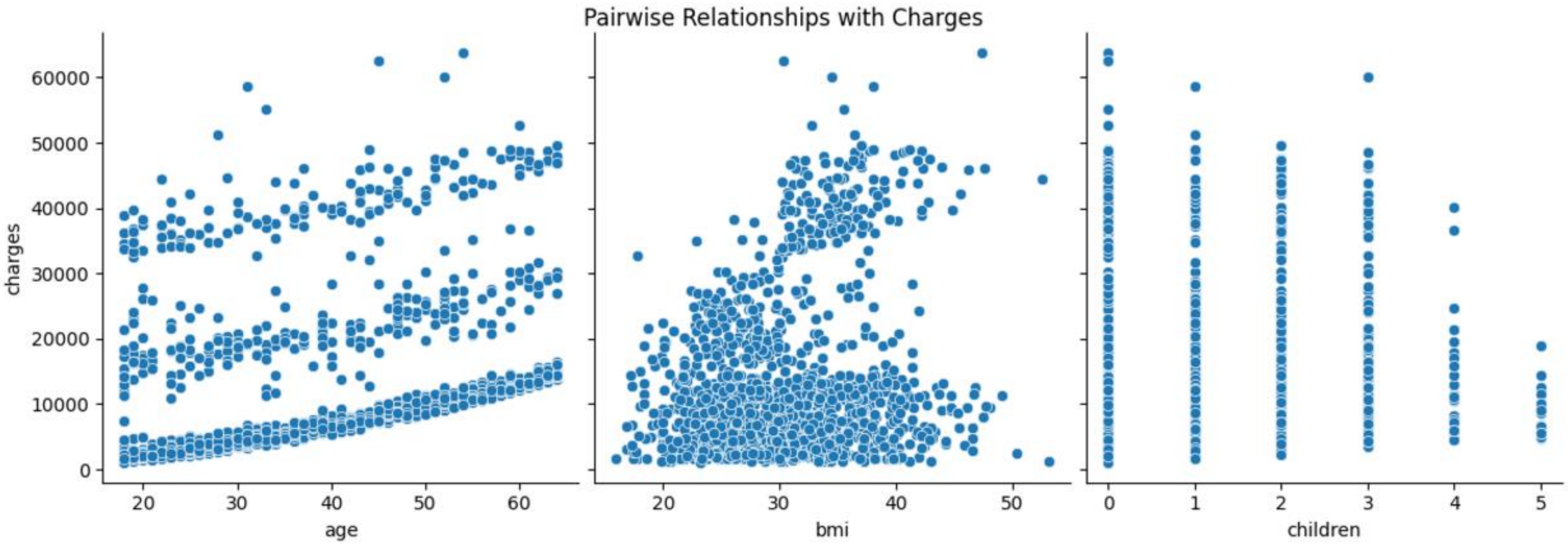
- **age:** Age of the beneficiary (numerical)
- **sex:** Gender of the individual (categorical: male/female)
- **bmi:** Body Mass Index, a measure of obesity (numerical)
- **children:** Number of dependents (numerical)
- **smoker:** Smoking status (categorical: yes/no)
- **region:** Residential area in the US (categorical: northeast, northwest, southeast, southwest)
- **charges:** The target variable representing medical insurance costs (numerical)

PROJECT STEPS

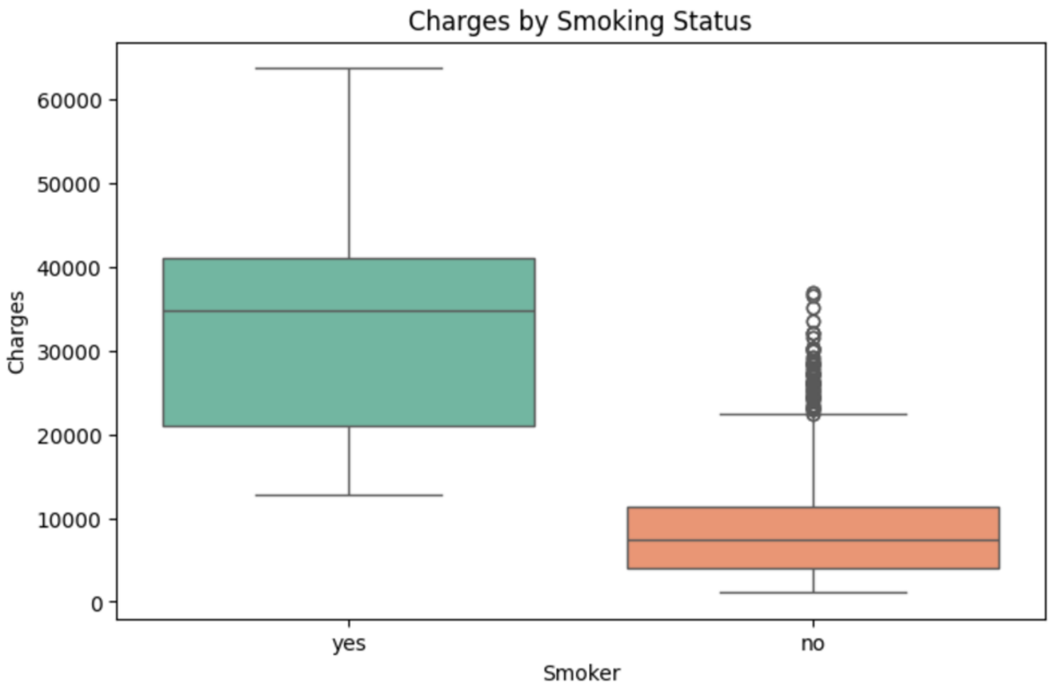
- Data Cleaning and EDA
- Model Building (3 Models)
- Model Evaluation

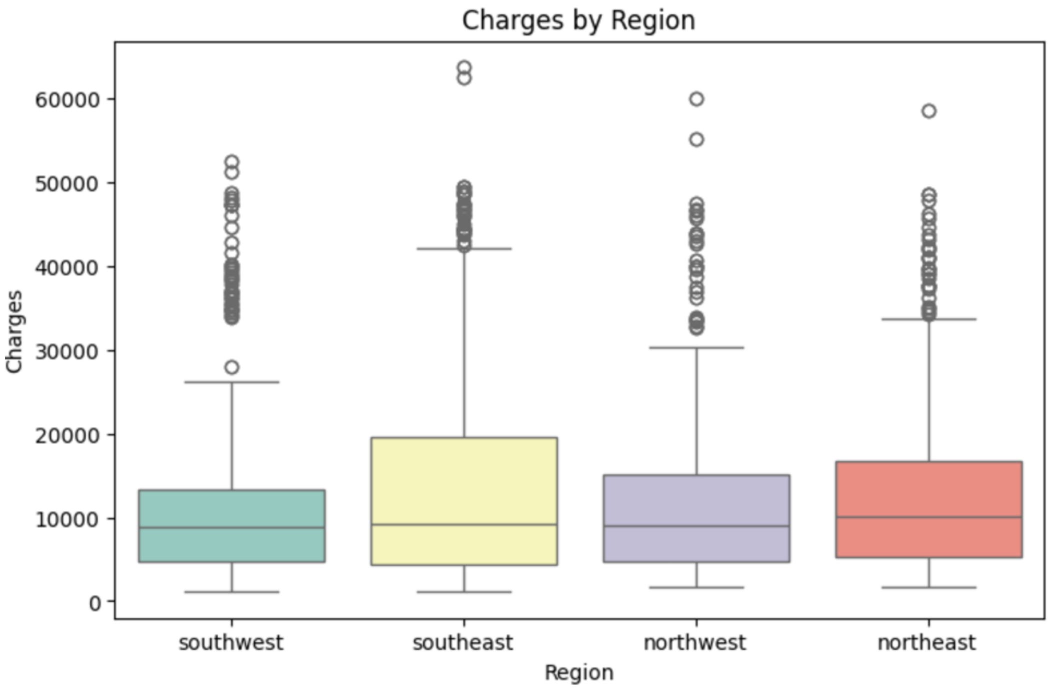


STEP 1 DATA CLEANING AND EDA

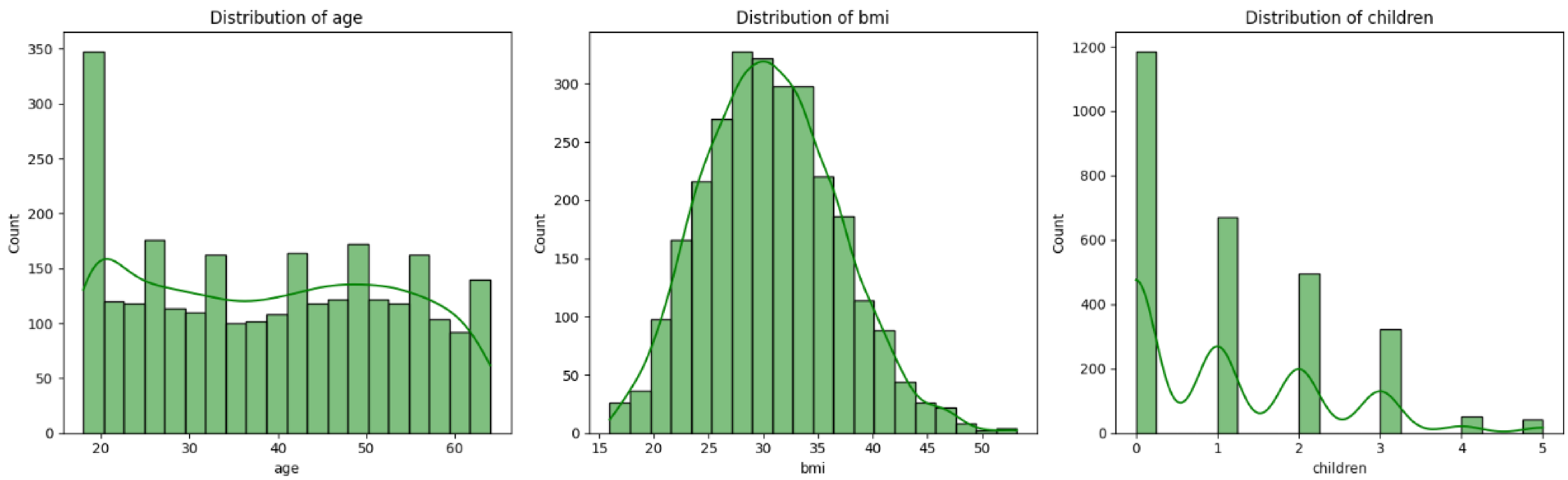


STEP 1 DATA CLEANING AND EDA





STEP 1 DATA CLEANING AND EDA



STEP 1 DATA CLEANING AND EDA

MODEL BUILDING

- Linear Regression Model
- Basic Decisión Tree
- Deep Learning (FFNN)

DATA PREPROCESSING

- Transformed all categorical variables (sex, smoker and region) into numeric vectors using One Hot Encoder.
- Scaled all numerical variables using StandardScaler()
- Separated the data into Train / Test with an 80 / 20 split.

```
# Define categorical and numerical columns
categorical_cols = ['sex', 'smoker', 'region']
numerical_cols = ['age', 'bmi', 'children']
```

```
# Define transformers
categorical_transformer = OneHotEncoder(drop='first')  # Avoid multicollinearity
numerical_transformer = StandardScaler()
```

LINEAR REGRESSION MODEL

```python
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error

# Step 1: Initialize the model
linear_model = LinearRegression()

# Step 2: Fit the model to the training data
linear_model.fit(X_train, y_train)

# Step 3: Make predictions
y_pred_train = linear_model.predict(X_train)
y_pred_test = linear_model.predict(X_test)

# Step 4: Evaluate the model
mse_train = mean_squared_error(y_train, y_pred_train)
mse_test = mean_squared_error(y_test, y_pred_test)

mae_train = mean_absolute_error(y_train, y_pred_train)
mae_test = mean_absolute_error(y_test, y_pred_test)

print("Linear Regression Performance:")
print(f"Training MSE: {mse_train:.2f}, Testing MSE: {mse_test:.2f}")
print(f"Training MAE: {mae_train:.2f}, Testing MAE: {mae_test:.2f}")
```

```
Linear Regression Performance:
Training MSE: 36787517.08, Testing MSE: 36782736.31
Training MAE: 4131.50, Testing MAE: 4289.93
```

DECISION TREE MODEL

```
from sklearn.tree import DecisionTreeRegressor

# Step 1: Initialize the model
decision_tree_model = DecisionTreeRegressor(random_state=52)

# Step 2: Fit the model to the training data
decision_tree_model.fit(X_train, y_train)

# Step 3: Make predictions
y_pred_train_tree = decision_tree_model.predict(X_train)
y_pred_test_tree = decision_tree_model.predict(X_test)

# Step 4: Evaluate the model
mse_train_tree = mean_squared_error(y_train, y_pred_train_tree)
mse_test_tree = mean_squared_error(y_test, y_pred_test_tree)
```

```
Decision Tree Regression Performance:
Training MSE: 235756.72, Testing MSE: 11133363.29
Training MAE: 28.55, Testing MAE: 708.06
```

DEEP LEARNING (FEED FORWARD NEURAL NETWORK)

```
# Step 1: Initialize the model
model = Sequential()
```

```
# Input layer
model.add(Dense(64, input_dim=X_train.shape[1], activation='relu'))
# Hidden layers
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.2))  # Dropout to prevent overfitting
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.2))  # Dropout to prevent overfitting
```

```
# Output layer
model.add(Dense(1, activation='linear'))
```

```
# Step 2: Compile the model
model.compile(optimizer=Adam(), loss='mean_squared_error', metrics=['mean_absolute_error'])
```

```
# Step 3: Train the model
early_stopping = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)
history = model.fit(X_train, y_train, epochs=100, batch_size=16, validation_split=0.3, verbose=1, callbacks=[early_stopping]
```

```
Deep Learning Model Performance:
Training MSE: 20891014.00, Testing MSE: 22092010.00
Training MAE: 2482.16, Testing MAE: 2652.52
```

Model	Training MSE	Testing MSE	Training MAE	Testing MAE
Linear Regression	3.67875e+07	3.67827e+07	4131.5	4289.93
Decision Tree Regression	235757	1.11334e+07	28.55	708.06
Deep Learning Model	2.0891e+07	2.2092e+07	2482.16	2652.52

MODEL EVALUATION

KEY INSIGHTS

- Linear Regression Simplicity
- Decision Tree Overfitting tendency
- Deep Learning Balance

FUTURE WORK

- Test on larger data (deep learning models tend to shine with larger data).
- Play more with other hyperparameters and additional features.
- Experiment with other DL architectures and Regularization Techniques

CONCLUSIONS AND SUGGESTIONS

THANK YOU

FIND THE FULL REPORT AND CODE ON GITHUB.