Introduction:

This project builds predictive models to determine whether a travel insurance policy will result in a claim. Using a dataset of past insurance sales and outcomes, we apply three supervised learning methods (Decision Tree, Logistic Regression, and a Neural Network) to solve a classification problem, predicting the categorical outcome “Claim” (Yes/No) from input features. The goal is to aid insurance providers in identifying high-risk policies. Accurate prediction of claims is important for setting premiums and managing risk. Since the target variable is a discrete label (Yes/No), this is a classification task.

Dataset Description:

The dataset contains 63,326 records and 10 input features plus the target feature. The features include categorical fields (e.g. Agency, Agency Type, Distribution Channel, Product Name, Destination, Gender) and numeric fields (Duration, Net Sales, Commission (in value), Age). The target column is Claim, indicating whether a claim was made. Claim has two categories (“Yes” or “No”), confirming a classification problem .

Correlation analysis was performed on numeric features to understand linear relationships. The heatmap of Pearson correlations reveals that Net Sales and Commission are strongly positively correlated, as expected because higher sales typically yield higher commission. Other numeric features have low mutual correlation. Crucially, the correlation between each feature and the binary claim label is very low, indicating no single numeric feature linearly predicts claims. In general, a correlation heatmap visually encodes pairwise correlation values by color intensity . Here, the heatmap shows no strong linear predictors of the claim outcome.
The bar chart for the target feature visually represents the frequency of each claim status in the dataset. It shows a prominent bar for "No" claims that is significantly taller than the bar for "Yes" claims. This stark difference in bar heights clearly illustrates the highly imbalanced nature of the dataset, where the majority of instances represent policies without a claim. A bar chart of class distribution highlights the overwhelming majority of “No” outcomes.
EDA insight: The data has no missing values except Gender. Some numeric features have unusual values: Net Sales has negative values (possibly refunds), Duration ranges widely, and Age has some more than 100. Categorical features are high-cardinality. In summary, the key EDA findings are the severe class imbalance, strong correlation only between Net Sales and Commission and otherwise weak direct patterns linking features to claims. All these suggest careful preprocessing and evaluation: we must use stratified splits to preserve class proportions and examine precision/recall (not just accuracy) due to the skewed target.

Dataset Pre-processing :

During preprocessing, the following issues were addressed
Nulls and Irrelevant Columns: The “Gender” column has a lot of missing values so it was dropped entirely. All other columns had no nulls.
Encoding Categorical Features: Categorical variables (Agency, Agency Type, Distribution Channel, Product Name, Destination) were transformed using one-hot encoding to convert them into numeric form (each category became a binary feature). This is necessary because scikit-learn models require numeric input.

Numeric Features and Scaling: The remaining features (Duration, Net Sales, Commission (in value), Age) are numeric but on very different scales. We applied standard scaling so that each numeric feature has mean 0 and unit variance. Feature scaling is important especially for Logistic Regression and Neural Network models to ensure balanced influence and faster convergence.
Encoding: The Claim target was encoded as binary (1 for “Yes”, 0 for “No”). We also encoded “Gender” to the same.
In summary, we removed the problematic Gender column, one-hot-encoded all categorical predictors, and scaled numeric columns. This produced a numeric input matrix suitable for modeling. These steps follow standard ML preprocessing practices: drop useless columns, encode categorical data, and normalize numeric features to avoid dominance by large-valued variables.

Dataset Splitting:

The data was split into training and test sets in a 70/30 ratio. Crucially, we used stratified sampling on the target label to ensure the training and test sets maintain the same class proportions as the original data. Given the extreme imbalance. Stratification can prevent it, for example, very few “Yes” instances in one set. Stratified splitting is recommended for imbalanced datasets to produce reliable evaluation. After splitting, the train set becomes significantly less imbalanced than before.

Model Training & Testing:

1. Decision Tree Classifier
   A Decision Tree classifier was trained on the preprocessed training set. The trained tree was then evaluated on the test set.

    a. Accuracy: The tree achieved 93.69% accuracy on the test set. However, this is misleadingly high due to class imbalance.
    b. Classification Report: The model’s precision and recall for the majority class (No) are both very high (≈0.97), but for the minority “Yes” class they are extremely low (≈0.11). The F1-score is also the same as precision and recall. These values appear in the classification report.
    c. Confusion Matrix: The confusion matrix confirms this: nearly all “No” instances were correctly identified but the model missed most “Yes” cases while making some false positive errors.
    d. ROC and AUC: The tree’s ROC curve is plotted in Figure 6. The AUC is only 0.5408, slightly above 0.5. An AUC of 0.5 corresponds to random guessing. So the Decision Tree effectively performs no better than chance on distinguishing the rare positive class.

2. Logistic Regression

Next, we trained a Logistic Regression model. The input data had been scaled, which is important for this gradient-based model.

a. Accuracy: The logistic model achieved 96.47% accuracy on the test set.
b. Classification Report: Despite the high accuracy, the logistic model predicted every instance as “No” (the majority class). This yields 0% recall and precision for the “Yes” class, the model fails to identify any actual claims. Hence the F1-score for “Yes” is 0. For the “No” class, precision and recall are 1.00. This reflects a trivial prediction strategy favored by maximizing accuracy under imbalance. The Logistic model also avoids predicting positives, leading to recall=0.

c. Confusion Matrix: The confusion matrix shows majority values for true negatives and 0 true positives. Rest are actual claims that appear as false negatives.
d. ROC and AUC: However, examining predicted probabilities reveals some signals. ROC performance is not that bad (AUC 0.6949) since its scores differentiate classes.

3. Neural Network:

A feedforward Neural Network (MLPClassifier with three hidden layers) was trained on the same data and inputs were scaled.

a. Accuracy: The network reached 0.96% test accuracy.
b. Classification Report: Similar to logistic, the network predicted almost all “No”.Here, TP=0 meaning it cannot correctly identify the claims. Thus recall for “Yes” is 0.00 along with precision and F1-score 0.01. “No” precision and recall remain the whole value.
c. Confusion Matrix: The network’s confusion matrix is nearly identical to logistic’s, except for one isolated correct claim.
d. ROC and AUC: The network’s ROC curve is similar to logistic’s. Its AUC is 0.74, which is not that bad.

Model Selection and Comparison:

Model performance metrics are summarized and compared. Bar chart shows the raw accuracy of each model on test data. Decision Tree has the lowest accuracy (93.69%), while Logistic (96.47%) and Neural Network (0.96%) are higher. However, accuracy alone is misleading with such imbalanced classes .
A more informative comparison uses precision, recall, and AUC. The Decision Tree achieved a very low recall on the minority class and correspondingly low precision, whereas Logistic and Neural Network had effectively zero recall and precision effectively 0. Thus, neither Decision tree nor logistic/Neural Network could reliably detect claims. Among these, the Logistic and Neural Network had the advantage in AUC (0.6949 and 0.74) versus 0.5408 for the Decision tree. Since a higher AUC indicates better model ranking performance, the Neural Network is best in this sense. The Decision Tree’s AUC 0.5408 is only marginally above 0.5, essentially equivalent to random guessing. Neural Network’s AUC of 0.74 is the highest, with the Logistic model close behind at 0.6949. ROC curves for all models are plotted, we see the curves for logistic (orange) and neural (red) are well above the random line, whereas the tree’s (yellow) is nearly diagonal. Confusion matrices (Figures 3–5) also contrast the models. The Decision Tree shows some correct positives while the others have almost none. Yet, even the Decision tree predicts so few positives that its recall is tiny. Precision vs recall trade-offs are clearly skewed by class imbalance.

Conclusion:

From the results, it is evident that although all three models (Decision Tree, Logistic Regression, and Neural Network) achieved high accuracy, they performed poorly in detecting the minority class (“Yes” for claims). This outcome underscores that accuracy is not a reliable metric in the presence of severe class imbalance. Precision, recall, F1-score, and especially AUC are better indicators in such scenarios.

Model Performance Summary:

- **Decision Tree** captured a few positive cases but still had very low recall and precision. It performed slightly better than random chance (AUC = 0.5408).

- **Logistic Regression** predicted no claims at all, resulting in recall and precision of 0 for the positive class. However, it had a better AUC (0.6949), indicating some ability to separate classes based on probability scores.

- **Neural Network** had the highest AUC (0.73), suggesting it was the best at distinguishing between classes, even if its predictions defaulted to the majority class.

Why the Models Performed Poorly:

The **extreme imbalance** in the target variable (claims are rare) led the models to favor predicting the majority class. Standard classifiers aim to maximize overall accuracy, which results in models ignoring the minority class if doing so increases their score. Features have **low linear correlation** with the target, making the learning task harder. Despite scaling and encoding, the input space still lacks clear signals for claim detection.

Challenges Faced:

Handling **imbalanced data** was the biggest challenge. Even with stratified splitting, the models struggled to learn useful patterns for the minority class. Choosing appropriate evaluation metrics and interpreting results beyond accuracy was critical. Preprocessing high-cardinality categorical features increased dimensionality, potentially affecting model performance and training time. Finding the right model complexity and avoiding overfitting or underfitting was non-trivial, especially with noisy features and little class signal.

Future Improvements:

Try **resampling techniques** like SMOTE (Synthetic Minority Over-sampling Technique) or undersampling the majority class. Use **cost-sensitive learning**, assigning higher penalties to misclassified positive instances. Explore **ensemble methods** like Random Forest or Gradient Boosting, which might capture subtle patterns better. Perform **feature engineering** or dimensionality reduction to reduce noise and improve signal quality. Consider using **deep learning** with dropout and tuning or **AutoML** frameworks for better optimization.

In conclusion, while the current models struggled with the imbalanced dataset, this project highlights key considerations in applying classification algorithms to real-world, skewed datasets. Thoughtful preprocessing, evaluation, and model tuning are crucial to achieving meaningful results.