

Problem

Predict individual health insurance premiums using demographic, lifestyle, and medical history data (1M records) to support faster and more consistent pricing decisions.

Data & Split

Used Insurance_Prediction table → 700k train, 200k validation, 100k production simulation to mirror real business rollout.

Approach

- Standardized text (lower/strip) to remove category noise
- Missing handling:
 - Numeric → median
 - Categorical → business defaults (No_Record / Unknown)
- Feature engineering:
 - Binary encoding (gender, smoker)
 - Ordinal encoding (exercise, coverage)
 - One-hot (medical + family history + occupation)
- Dropped **region** after consensus feature irrelevance across Linear, Lasso, RF.
- Built baseline (**Linear Regression**), regularized (**Lasso**), and non-linear (**Random Forest**) and compared on the same pipeline.
- Ensured **identical preprocessing** in training, batch prediction, and Flask API (industry deployment standard).

Model Comparison (from your runs)

- **Linear Regression:** $R^2 \approx 0.9851$, training **0.32s**, inference **0.013s**, size **~0.8 KB**
- **Lasso:** $R^2 \approx 0.9851$, training **0.38s**, inference **0.007s**, size **~0.75 KB**
- **Random Forest:** $R^2 \approx 0.9869$, training **~98s**, inference **~8.39s**, size **~1.1 TB → GB scale**

Final Model Chosen → Lasso (Production-Grade Tradeoff)

- Same accuracy as Linear, **much smaller**, stable with correlated features, auto-removes weak signals.
- Random Forest gave +0.0018 R^2 but **massive cost in latency and storage** → not practical for real-time premium APIs.

Key Feature Insights

- Highest impact: **smoker, coverage_level, heart disease history**
- Weak/removed: **region**, several occupation dummies (low importance across models).

Business Impact

- Automated premium estimation with **sub-millisecond inference**, enabling real-time quote generation and consistent pricing.

Risk / Failure Modes

- New unseen categories (new medical/occupation values) → handled via schema alignment.
- Data drift (e.g., sudden smoker distribution change) could degrade accuracy → needs monitoring.

Post-Deployment Monitoring

- Track **RMSE**, prediction distribution vs training, and % of unseen categories.

Why Lasso despite RF slightly higher score?

Because **same accuracy + 10,000× smaller + 1000× faster** → best **accuracy-latency-cost** balance for production.

Cost of wrong prediction

- Underpricing → financial loss
- Overpricing → customer churn → revenue loss

Most sensitive feature

- Removing **smoker** causes largest performance drop (highest coefficient magnitude & RF importance).

Behavior with new cities

- **region** removed → model is robust to unseen geography.

Deployment Artifacts Delivered

- Reusable preprocessing pipeline
- train.py, predict.py batch scoring
- Flask API for real-time premium prediction
- Saved model + scaler + feature schema for consistency