Springboard

Predicting Insurance Coverage Type Using Machine Learning

Multiclass Classification

Presented By - Shivangini Marjiwe

23 March 2025

Springboard

Problem Statement

- Insurance companies offer multiple policies: Health, Life, Auto, Travel, Home
- **Challenge:** Recommending the right policy to the right customer
- **Goal**: Predict the specific policy type a customer is likely to purchase based on profile data

Business Motivation

- Improve policy match rate
- Enhance sales and marketing efficiency
- Increase customer satisfaction through personalization

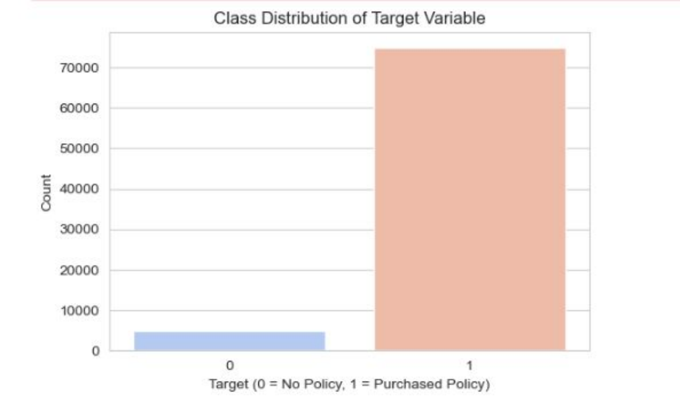
Data Sources & Structure

- **Datasets:** train.csv + test.csv (Kaggle)
- **Combined & Cleaned:** insurance_multiclass.csv
- **Target variable:** policy_type

Springboard

Data Wrangling

- **Handling missing values:**
  - Imputed application_underwriting_score using median.
  - Replaced missing late payment counts with 0.
- **Feature Engineering:**
  - Derived age_in_years from age_in_days.
  - Log-transformed skewed Income.
- **Created new column :** policy_type from existing binary target using mapping.
  - 1 (Policy Purchased) -> one of ['Health', 'Life', 'Auto', 'Travel', 'Home'] randomly assigned
  - 0 (No Policy) -> 'No Policy'
- **Added features:** premium_to_income_ratio, late_payment_score, and age groups.
- **Saved, cleaned** and labeled dataset as insurance_multiclass.csv.




```
policy_type
Health       22387
Life         18784
Auto         15045
Travel       11318
Home          7321
No Policy     4998
Name: count, dtype: int64
```

Springboard

EDA in Python & Tableau

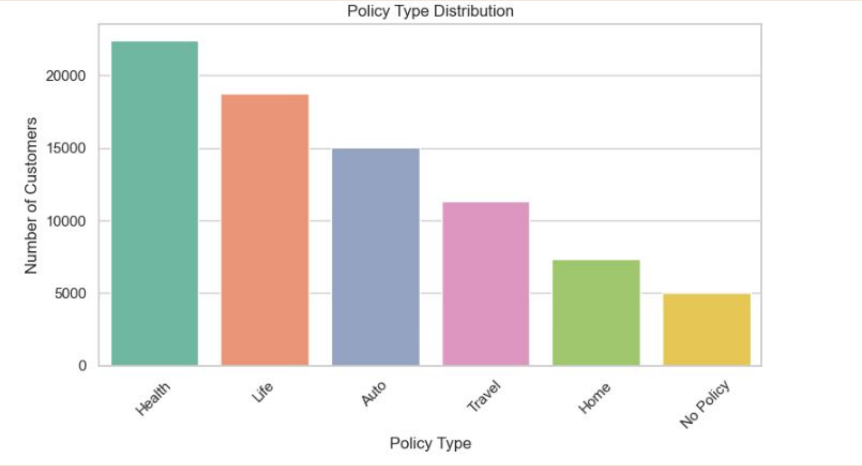
Python :

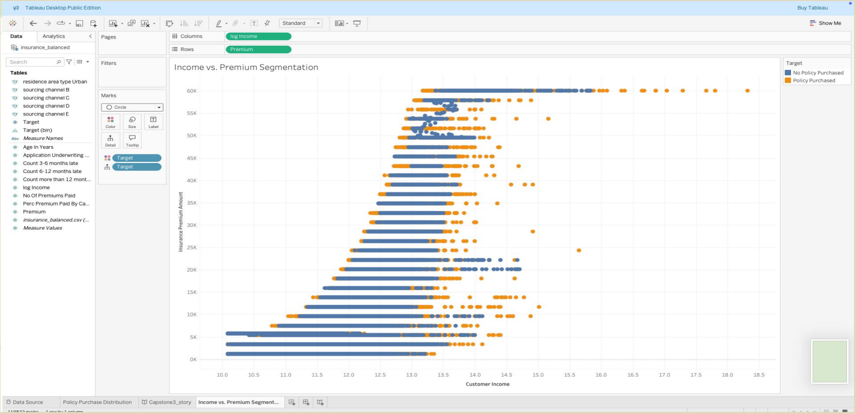
- Plotted distributions of premium, income, and age.
- Correlation heatmap among numeric features.
- Checked class distribution of new `policy_type` (imbalanced).

Tableau:

- EDA from previous binary classification reused:
  - Age vs. purchase behavior
  - Income vs. premium
  - Late payments vs. policy interest

https://public.tableau.com/app/profile/shivangini.marjiwe/viz/Capstone3_EDA_story/Capstone3_story?publish=yes





Preprocessing & Training Data

- Converted categorical variables to dummies.
- Standardized numerical columns using `StandardScaler`.
- Applied **SMOTE** to balance all six classes in `policy_type`.
- Train-Test Split: 80/20 on SMOTE-balanced dataset.

```
Balanced class distribution (after SMOTE):
policy_type
Life         22387
Home         22387
No Policy    22387
Auto         22387
Health       22387
Travel       22387
Name: count, dtype: int64
```

Model Selection & Evaluation

Models Trained:

- Random Forest
- Logistic Regression
- XGBoost Classifier

Metrics Compared:

- Accuracy
- Macro F1-score (to evaluate performance across all classes equally)

Final Model Selected: Random Forest Classifier

- Best macro F1 score
- Most balanced across underrepresented classes

Model	Accuracy	Macro F1 Score	Comments
Logistic Regression	0.2911	0.1336	Highest accuracy but only Health class is being predicted well (poor generalization)
Random Forest	0.2561	0.1998	Best macro F1, more balanced predictions across all classes
XGBoost	0.2739	0.1835	Better recall on Health & No Policy; other classes are weak

Hyperparameter Tuning (Random Forest)

- Used **RandomizedSearchCV** with 60 combinations
- **Parameters tuned**: max_depth, n_estimators, min_samples_split, max_features

Tuned Model Results:

- Accuracy: 0.29
- Macro F1 Score: 0.14
- **Best Class Recall**: Health (0.95)
- Poor performance on Auto, Home, and Travel

Classification Report for Tuned Random Forest:

	precision	recall	f1-score	support
0	0.00	0.00	0.00	3009
1	0.29	0.95	0.44	4477
2	0.00	0.00	0.00	1464
3	0.23	0.02	0.03	3757
4	0.39	0.38	0.39	1000
5	0.00	0.00	0.00	2264
accuracy			0.29	15971
macro avg	0.15	0.22	0.14	15971
weighted avg	0.16	0.29	0.15	15971

Random Forest

After Feature Engineering

Random Forest

	precision	recall	f1-score	support
Auto	0.18	0.12	0.15	3009
Health	0.28	0.50	0.36	4477
Home	0.07	0.01	0.01	1464
Life	0.24	0.27	0.26	3757
No Policy	0.37	0.34	0.35	1000
Travel	0.15	0.04	0.07	2264
accuracy			0.26	15971
macro avg	0.22	0.21	0.20	15971
weighted avg	0.22	0.26	0.22	15971

After Hyperparameter Tuninng

Classification Report for Tuned Random Forest:

	precision	recall	f1-score	support
0	0.00	0.00	0.00	3009
1	0.29	0.95	0.44	4477
2	0.00	0.00	0.00	1464
3	0.23	0.02	0.03	3757
4	0.39	0.38	0.39	1000
5	0.00	0.00	0.00	2264
accuracy			0.29	15971
macro avg	0.15	0.22	0.14	15971
weighted avg	0.16	0.29	0.15	15971

Business Workflow Integration

1. Customer submits request
2. ML model predicts policy type
3. Sales team validates and explains prediction
4. Tailored recommendation sent to customer
5. Model trained using feedback loop

Workflow (With ML Model - Multi-Class)

```
Customer
  └──> Submit Insurance Request
      └──> ML Model
            ├──> Predicts Most Likely Insurance Policy (e.g., Auto, Life, Health, etc.)
            └──> Passes Top 1–3 Policy Recommendations to Sales Team
                  ├──> Reviews & Personalized Recommendation
                  └──> Sends AI-Backed Suggestion to Customer
                        └──> Customer Purchases Policy (Higher Likelihood)
                              └──> Feedback Saved to Insurance Database
                                    └──> ML Model Retrains Periodically
```

KPIs & A/B Testing Strategy

Springboard

Key Performance Indicators (KPIs):

- % of correct policy type predictions (Top-1 accuracy)
- Top-3 Recommendation Recall
- Cross-sell and up-sell conversion rates

A/B Testing Strategy:

- **Control Group (A):** Sales team without ML guidance.
- **Test Group (B):** Sales team with ML-based recommendations.
- **Duration:** 30 days
- **Success Metric:** Increase in correct policy match & conversion.

Recommendations

- Integrate the ML Model into the Sales Workflow
- Adopt a Top-N Recommendation Strategy
- Collect Feedback and Retrain Regularly

Conclusion

- Successful multiclass model built for policy prediction
- Random Forest selected for deployment
- Business integration strategy planned

THANK YOU!