Rapport projet science de données

Contexte & Objectif:

Le but de ce projet de sciences de données est de construire un modèle de classification binaire capable de prédire si un assuré fera une demande d'indemnisation (outcome) au cours de la période d'assurance. Ainsi nous disposons d'un jeu de données de la compagnie « On the Road » et nous allons étape par étape : Importer et exporter les données, traiter les valeurs manquantes et aberrantes, encoder et normaliser les variables, entraîner plusieurs classifieurs, les comparer et sauvegarder le meilleur modèle.

Traitement des données

Le projet a démarré par une phase d'exploration où nous avons rapidement constaté deux défis majeurs : des valeurs manquantes sur le credit_score et l'annual_mileage (environ 10 % chacune) et quelques données aberrantes (par exemple des assurés déclarant plus de douze enfants). Plutôt que de supprimer des lignes et de risquer de biaiser la population, donc la fiabilité de notre modèle, nous avons choisi la médiane pour ces deux variables. Puis, nous avons plafonné les données aberrantes (dans le nombre d'enfants, kilométrage annuel, infractions pour excès de vitesse et accidents antérieurs) à cette même valeur médiane. Pour les valeurs qui n'étaient pas des entiers ou des float, nous avons utilisé la méthode .mode de numpy, qui remplace la valeur par celle la plus couramment utilisée sur cette colonne. Cette stratégie d'imputation assure la robustesse du modèle face aux données aberrantes et préserve l'intégrité statistique du jeu de données.

Vient ensuite la préparation des features : les colonnes qualitatives (driving_experience, education, income, vehicle_year, vehicle_type) ont été transformées en numérique grâce à LabelEncoder, car elles comportent des modalités ordonnées ou limitées. Pour ne pas fausser les calculs de distances ou la convergence de l'algorithme, nous avons appliqué une standardisation (StandardScaler) sur l'ensemble des variables numériques, garantissant ainsi que toutes contribuent de façon équilibrée à la modélisation.

Évaluation et comparaison avec d'autres classifieurs :

La division entre, apprentissage, et test, a été réalisée avec train_test_split (70 % / 30 %, random_state=42), afin d'évaluer la généralisation sans fuite d'information.

Trois classifieurs ont été configurés pour couvrir différentes approches :

une régression logistique (LogisticRegression avec max_iter=1000 et régularisation L2 par défaut) pour son interprétabilité ;

un Perceptron (max_iter=1000) pour tester une méthode de gradient itératif basique ;

un K plus proches voisins (KNeighborsClassifier, k=5) pour mesurer l'impact de la similarité locale.

Les performances obtenues montrent que la régression logistique atteint près de 81 % d'accuracy, un recall autour de 76 % et une précision proche de 72 %, traduction d'un bon compromis entre détection des sinistres et maîtrise du taux de fausses alertes. Le Perceptron est un peu en retrait (≈ 78 % d'accuracy) du fait d'une convergence plus instable, tandis que le KNN se positionne légèrement au-dessus (≈ 82 % d'accuracy) mais souffre de son coût de calcul en production.

Pour approfondir, on envisagera un GridSearchCV sur l'hyper paramètre C de la régression logistique et sur k pour le KNN, ou encore l'introduction de SMOTE pour corriger un éventuel déséquilibre de classes. L'analyse de feature importance (par coefficients ou techniques de sélection récursive) permettra de simplifier le modèle en ne retenant que les variables les plus prédictives. Enfin, la validation via ROC AUC et la calibration des probabilités renforceront la fiabilité du score prédictif lorsqu'on l'intégrera en production.

Conclusion

Ce projet a été très enrichissant, il nous a tous deux appris à traiter efficacement des données ainsi que l'apprentissage d'un modèle. Les résultats obtenus sont très satisfaisants pour un premier modèle.