Experiment Notebook

Setup Environment

In [1]:
```
# DO NOT MODIFY THE CODE IN THIS CELL
!pip install -q utstd

from utstd.folders import *
from utstd.ipyrenders import *

at = AtFolder(
    course_code=36106,
    assignment="AT1",
)
at.run()

import warnings
warnings.simplefilter(action='ignore')
```

```
ERROR: Could not install packages due to an OSError: [WinError 5] 拒绝访问。: 'C:\\Users\\brohao\\AppData\\Local\\Programs\\Python\\Python311\\Lib\\site-packages\\~3learn\\.libs\\msvcp140.dll'
Consider using the `--user` option or check the permissions.


[notice] A new release of pip available: 22.3.1 -> 25.2
[notice] To update, run: python.exe -m pip install --upgrade pip
You can now save your data files in: c:\Users\brohao\Desktop\UTS\36106\AT1\36106\assignment\AT1\data
```

Student Information

In [2]:
```
student_name = "Jiayu Hao"
student_id = "25948860"
```

In [3]:
```
# DO NOT MODIFY THE CODE IN THIS CELL
print_tile(size="h1", key='student_name', value=student_name)
```

student_name

Jiayu Hao

In [4]:
```
# DO NOT MODIFY THE CODE IN THIS CELL
print_tile(size="h1", key='student_id', value=student_id)
```

student_id

25948860

---

0. Python Packages

0.a Install Additional Packages

> If you are using additional packages, you need to install them here using the command: `! pip install <package_name>`

In [5]:
```
!pip install scikit-learn
```

```
Requirement already satisfied: scikit-learn in c:\users\brohao\appdata\local\programs\python\python311\lib\site-packages (1.6.1)
Requirement already satisfied: numpy>=1.19.5 in c:\users\brohao\appdata\local\programs\python\python311\lib\site-packages (from scikit-learn) (2.3.2)
Requirement already satisfied: scipy>=1.6.0 in c:\users\brohao\appdata\local\programs\python\python311\lib\site-packages (from scikit-learn) (1.16.1)
Requirement already satisfied: joblib>=1.2.0 in c:\users\brohao\appdata\local\programs\python\python311\lib\site-packages (from scikit-learn) (1.5.1)
Requirement already satisfied: threadpoolctl>=3.1.0 in c:\users\brohao\appdata\local\programs\python\python311\lib\site-packages (from scikit-learn) (3.6.0)
[notice] A new release of pip available: 22.3.1 -> 25.2
[notice] To update, run: python.exe -m pip install --upgrade pip
```

0.b Import Packages

In [6]:
```
# DO NOT MODIFY THE CODE IN THIS CELL
import pandas as pd
import altair as alt
```

In [7]:
```
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_absolute_error
```

A. Experiment Description

In [8]:
```
# DO NOT MODIFY THE CODE IN THIS CELL
experiment_id = "3"
print_tile(size="h1", key='experiment_id', value=experiment_id)
```

experiment_id

3

In [9]:
```
# Present the hypothesis you want to test, the question you want to answer or the insight you
# Explain the reasons why you think it is worthwhile considering it
experiment_hypothesis = """
The hypothesis is that KNN can capture local similarity patterns that may improve premium pred
The question is whether customers with similar profiles pay similar premiums and if KNN can us
This is worthwhile because it tests nonlinear relations and provides chance for locality-based
"""
```

In [10]:
```
# DO NOT MODIFY THE CODE IN THIS CELL
print_tile(size="h3", key='experiment_hypothesis', value=experiment_hypothesis)
```

experiment_hypothesis

The hypothesis is that KNN can capture local similarity patterns that may improve premium prediction compared to linear models. The question is whether customers with similar profiles pay similar premiums and if KNN can use this to lower validation MAE. This is worthwhile because it tests nonlinear relations and provides chance for locality-based pricing.

In [11]:
```
experiment_expectations = """
Detail what will be the expected outcome of the experiment. If possible, estimate the goal you
List the possible scenarios resulting from this experiment.
The expected outcome is that a well-scaled KNN with tuned k, p, and weights will match or slig
We will read results by tracking MAE across (k,p,weights) if MAE falls as k increases (then pl
if distance weighting beats uniform, nearer neighbors carry more signal; if p=1 ≈ p=2, the met
Possible scenarios: (1) MAE improves—KNN adds nonlinear value and is a candidate for deployme
(2) MAE is similar—use KNN as an interpretability/benchmark but keep linear models;
(3) MAE is worse—high-dimensional distance noise dominates.
"""
```

In [12]:
```
# DO NOT MODIFY THE CODE IN THIS CELL
print_tile(size="h3", key='experiment_expectations', value=experiment_expectations)
```

experiment_expectations

Detail what will be the expected outcome of the experiment. If possible, estimate the goal you are expecting. List the possible scenarios resulting from this experiment. The expected outcome is that a well-scaled KNN with tuned k, p, and weights will match or slightly improve validation MAE versus Ridge/Lasso (target MAE: 126-128). We will read results by tracking MAE across (k,p,weights) if MAE falls as k increases (then plateaus), smoothing helps; if distance weighting beats uniform, nearer neighbors carry more signal; if p=1 ≈ p=2, the metric is not the bottleneck. Possible scenarios: (1) MAE improves—KNN adds nonlinear value and is a candidate for deployment; (2) MAE is similar—use KNN as an interpretability/benchmark but keep linear models; (3) MAE is worse—high-dimensional distance noise dominates.

B. Feature Selection

In [13]:
```
# DO NOT MODIFY THE CODE IN THIS CELL
# Load data
try:
  X_train = pd.read_csv(at.folder_path / 'X_train.csv')
  y_train = pd.read_csv(at.folder_path / 'y_train.csv')

  X_val = pd.read_csv(at.folder_path / 'X_val.csv')
  y_val = pd.read_csv(at.folder_path / 'y_val.csv')

  X_test = pd.read_csv(at.folder_path / 'X_test.csv')
  y_test = pd.read_csv(at.folder_path / 'y_test.csv')
except Exception as e:
  print(e)
```

In [14]:
```
# Selected features come from the Lasso-selected set
selected_feats = [
    "payment_method_0","second_driver_0","distribution_channel_1","vehicle_value",
    "policy_type_3","total_claims_number_ratio","car_age","lapsed_policies",
    "driving_experience","total_claims_number_in_history","seniority",
    "current_policies_held","vehicle_weight","vehicle_length",
    "total_claims_number_in_current_year","vehicle_horsepower",

    # some features may be added to selected_feats

    # "gender_m","vehicle_fuel_type_D","vehicle_cylinder","max_policies_held"
]
features_list = selected_feats
```

In [15]:
```
print("Using features:", len(selected_feats), selected_feats[:10], "...")
```

```
Using features: 16 ['payment_method_0', 'second_driver_0', 'distribution_channel_1', 'vehicle_
value', 'policy_type_3', 'total_claims_number_ratio', 'car_age', 'lapsed_policies', 'driving_e
xperience', 'total_claims_number_in_history'] ...
```

In [16]:
```
# Provide a rationale on why you are selected these features but also why you decided to remov
feature_selection_explanations = """
The chosen features come from the Lasso-selected set, which removes weak or noisy variables ar
Removed features include identifiers, high-cardinality text, raw dates, and those consistently
"""
```

In [17]:
```
# DO NOT MODIFY THE CODE IN THIS CELL
print_tile(size="h3", key='feature_selection_explanations', value=feature_selection_explanati
```

feature_selection_explanations

The chosen features come from the Lasso-selected set, which removes weak or noisy variables and reduces dimensionality, important for distance-based models like KNN. Removed features include identifiers, high-cardinality text, raw dates, and those consistently shrunk to zero by Lasso, as they add noise and increase the risk of the curse of dimensionality.

C. Train Machine Learning Model

C.1 Import Algorithm

In [18]:
```
# Provide some explanations on why you believe this algorithm is a good fit
algorithm_selection_explanations = """
KNN is a good fit because it is a distance-based method that can capture local nonlinear patte
It is interpretable through its neighbors, and serves as a useful benchmark against regularize
"""
```

In [19]:
```
# DO NOT MODIFY THE CODE IN THIS CELL
print_tile(size="h3", key='algorithm_selection_explanations', value=algorithm_selection_expla
```

algorithm_selection_explanations

KNN is a good fit because it is a distance-based method that can capture local nonlinear patterns missed by linear models. It is interpretable through its neighbors, and serves as a useful benchmark against regularized regression.

C.2 Set Hyperparameters

In [20]:
```
param_k = [5, 10, 20, 30, 50]      # Small k more flexible and large k more stable
param_p = [1, 2]                   # 1=Manhattan 2=Euclidean
param_w = ["uniform", "distance"]  # Whether neighbors contribute through distance
```

In [21]:
```
# Explain why you are tuning these hyperparameters
hyperparameters_selection_explanations = """
We tune these hyperparameters because they control how KNN learns from neighbors.
The number of neighbors (k) sets smoothness, with small k more flexible and large k more stabl
The distance metric (p) changes how similarity is measured, with 1 for Manhattan and 2 for Euc
The weights option decides whether all neighbors contribute equally or closer ones have more i
"""
```

In [22]:
```
# DO NOT MODIFY THE CODE IN THIS CELL
print_tile(size="h3", key='hyperparameters_selection_explanations', value=hyperparameters_sel
```

hyperparameters_selection_explanations

We tune these hyperparameters because they control how KNN learns from neighbors. The number of neighbors (k) sets smoothness, with small k more flexible and large k more stable. The distance metric (p) changes how similarity is measured, with 1 for Manhattan and 2 for Euclidean. The weights option decides whether all neighbors contribute equally or closer ones have more influence.

C.3 Fit Model

In [23]:
```
Xtr = X_train[features_list]
Xva = X_val[features_list]

results = []
for k in param_k:
    for p in param_p:
        for w in param_w:
            knn = KNeighborsRegressor(n_neighbors=k, p=p, weights=w)
            knn.fit(Xtr, y_train)
            yva_pred = knn.predict(Xva)
            val_mae = mean_absolute_error(y_val, yva_pred)
            results.append((k, p, w, val_mae))

# Sort KNN validation MAE and print 10 better row
results_sorted = sorted(results, key=lambda x: x[3])
print("KNN Validation MAE (sorted):")
print(" k | p |  weights  | Val MAE")
for k, p, w, mae in results_sorted[:10]:
    print(f"{k:2} | {p} | {w:9} | {mae:7.2f}")

best_k, best_p, best_w, best_mae = results_sorted[0]
print("\nBest params:", {"n_neighbors": best_k, "p": best_p, "weights": best_w})
print("Best Val MAE:", round(best_mae, 2))
```

```
KNN Validation MAE (sorted):
 k | p |  weights  | Val MAE
 5 | 1 | distance  |  128.58
 5 | 1 | uniform   |  128.69
10 | 1 | distance  |  128.84
10 | 1 | uniform   |  128.94
 5 | 2 | distance  |  129.00
 5 | 2 | uniform   |  129.09
20 | 1 | distance  |  129.26
20 | 1 | uniform   |  129.37
30 | 1 | distance  |  129.51
10 | 2 | distance  |  129.54

Best params: {'n_neighbors': 5, 'p': 1, 'weights': 'distance'}
Best Val MAE: 128.58
```

D. Model Evaluation

D.1 Model Technical Performance

In [24]:
```python
# Provide some explanations on model performance
model_performance_explanations = """
The model performance is read by tracking validation MAE across k, p, and weights.
If MAE falls as k grows, the model gains from smoothing;
if distance weighting is better than uniform, closer neighbors hold stronger signals;
if p=1 and p=2 give similar results, the distance metric is not limiting.
Compared to baselines,
if KNN MAE is close to Lasso or Ridge, the data is mostly linear;
if worse, KNN is affected by high-dimensional distance noise.
"""
```

In [25]:
```python
# DO NOT MODIFY THE CODE IN THIS CELL
print_tile(size="h3", key='model_performance_explanations', value=model_performance_explanati
```

model_performance_explanations

The model performance is read by tracking validation MAE across k, p, and weights. If MAE falls as k grows, the model gains from smoothing; if distance weighting is better than uniform, closer neighbors hold stronger signals; if p=1 and p=2 give similar results, the distance metric is not limiting. Compared to baselines, if KNN MAE is close to Lasso or Ridge, the data is mostly linear; if worse, KNN is affected by high-dimensional distance noise.

D.2 Business Impact from Current Model Performance

In [26]:
```python
business_impacts_explanations = """
Higher MAE means larger average pricing errors.
Overpricing increases the risk of customer churn, while underpricing raises the risk of losses
"""
```

In [27]:
```python
# DO NOT MODIFY THE CODE IN THIS CELL
print_tile(size="h3", key='business_impacts_explanations', value=business_impacts_explanation
```

business_impacts_explanations

Higher MAE means larger average pricing errors. Overpricing increases the risk of customer churn, while underpricing raises the risk of losses from claims.

E. Conclusion

In [28]:
```
experiment_outcome = "Hypothesis Rejected"
```

In [29]:
```
# DO NOT MODIFY THE CODE IN THIS CELL
print_tile(size="h2", key='experiment_outcomes_explanations', value=experiment_outcome)
```

experiment_outcomes_explanations

Hypothesis Rejected

In [30]:
```
# <Student to fill this section and then remove this comment>
experiment_results_explanations = """
The KNN experiment shows that local similarity patterns did not improve premium prediction, wi
This confirms that distance-based methods struggle with high-dimensional data, even when featu
The insight is that KNN serves as a useful nonlinear benchmark but is not suitable for product
The next steps should focus on feature engineering with interactions and bins, ElasticNet to b
Data quality improvements can also provide medium uplift.
If a model must be shipped,  KNN should not be deployed now.
"""
```

In [31]:
```
# DO NOT MODIFY THE CODE IN THIS CELL
print_tile(size="h2", key='experiment_results_explanations', value=experiment_results_explana
```

experiment_results_explanations

The KNN experiment shows that local similarity patterns did not improve premium prediction, with the best validation MAE (128.6) still worse than Lasso. This confirms that distance-based methods struggle with high-dimensional data, even when features are filtered. The insight is that KNN serves as a useful nonlinear benchmark but is not suitable for production. The next steps should focus on feature engineering with interactions and bins, ElasticNet to balance sparsity and stability, and tree-based models to capture nonlinear effects. Data quality improvements can also provide medium uplift. If a model must be shipped, KNN should not be deployed now.