GROUP 4

AUSTIN CRASH DATA

HUNG TRAN, STEVEN TRAN, LUCA COMBA

AGENDA

Introduction

Data Cleansing + Exploratory Analysis

Feature Selection + ML Methodology

GridSearch + Results

Conclusion

DATASET

THE DATASET INCLUDES RECORDS OF TRAFFIC ACCIDENTS IN *AUSTIN,* TEXAS.

FROM 2010 UNTIL TODAY, THE CITY OF *AUSTIN* COLLECTED 216,088 INSTANCES OF CAR ACCIDENTS.

THE DATA IS AVAILABLE AT HTTPS://CATALOG.DATA.GOV/DATASET/VISION-ZERO-CRASH-REPORT-DATA

VISION ZERO PROJECT HTTPS://VISIONZERO.AUSTIN.GOV/VIEWER/

DATASET

The dataset originally contains **45 features**.

Some of the important features were:

- Latitude & Longitude
- Crash timestamp (US/Central)
- Crash Speed Limit (MPH)
- Crash Severity (A scale from 0 to 5)
- Total Injury Count
- Total Death Count
- Model of units involved in crash
- Estimated Total Comprehensive Cost

A full description is available at https://data.austintexas.gov/Transportation-and-Mobility/Austin-Crash-Report-Data-Crash-Level-Records/y2wy-tgr5/about_data

COMPREHENSIVE CRASH COSTS

IS THE ECONOMIC AND QUALITY OF LIFE COSTS ASSOCIATED WITH ALL INJURIES SUSTAINED IN THE CRASH.

ECONOMIC COSTS

- MEDICAL BILLS,
- LOST WAGES

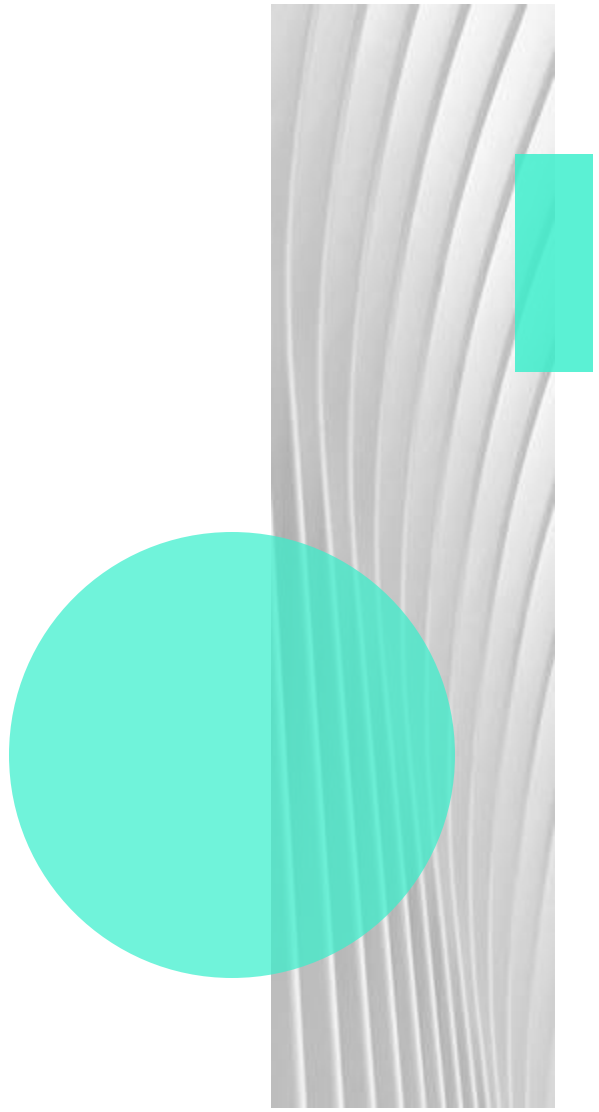
QUALITY OF LIFE COSTS

- RESULTING FROM TRAFFIC CRASHES (E.G., PHYSICAL PAIN, EMOTIONAL SUFFERING)
- WHICH ARE **INHERENTLY IMMEASURABLE**

LEARN MORE AT HTTPS://WWW.AUSTINTEXAS.GOV/CRASHCOSTS

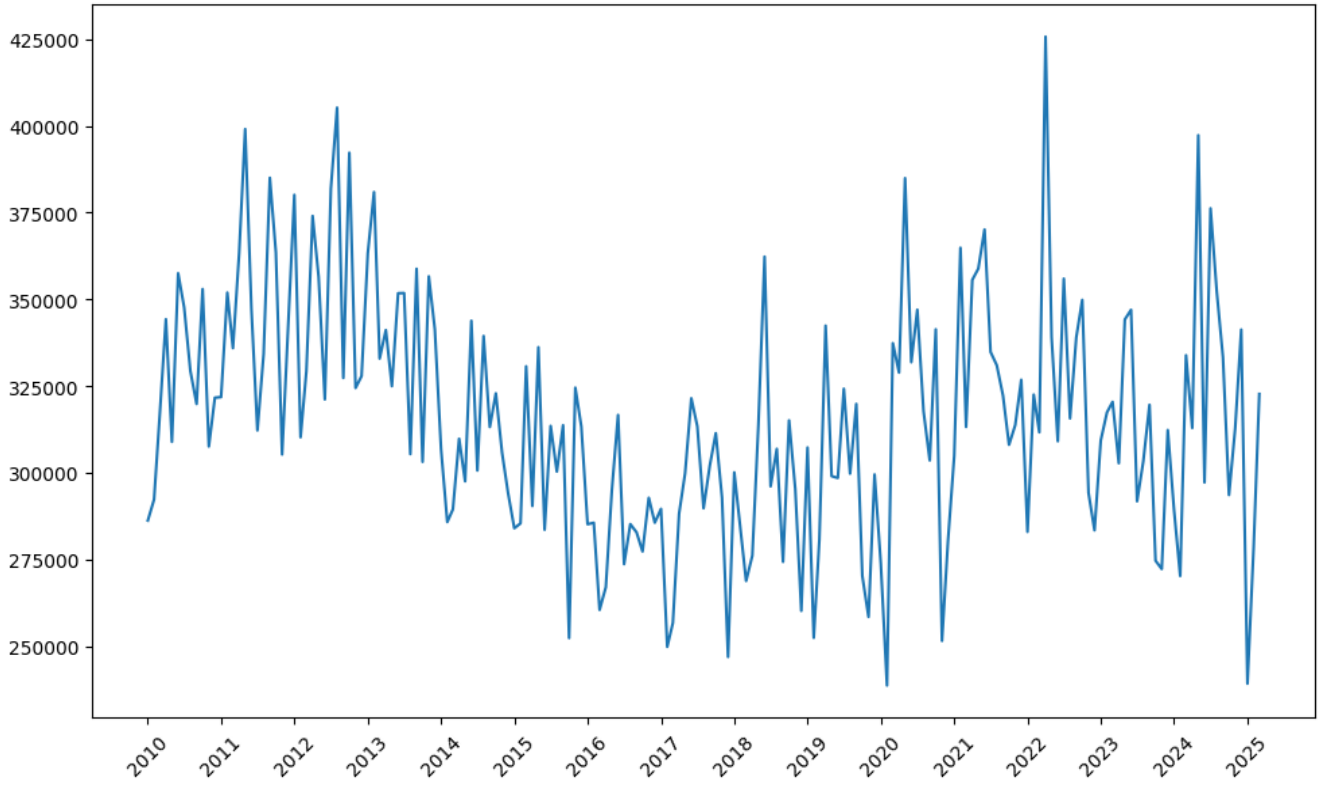
DATA CLEANSING

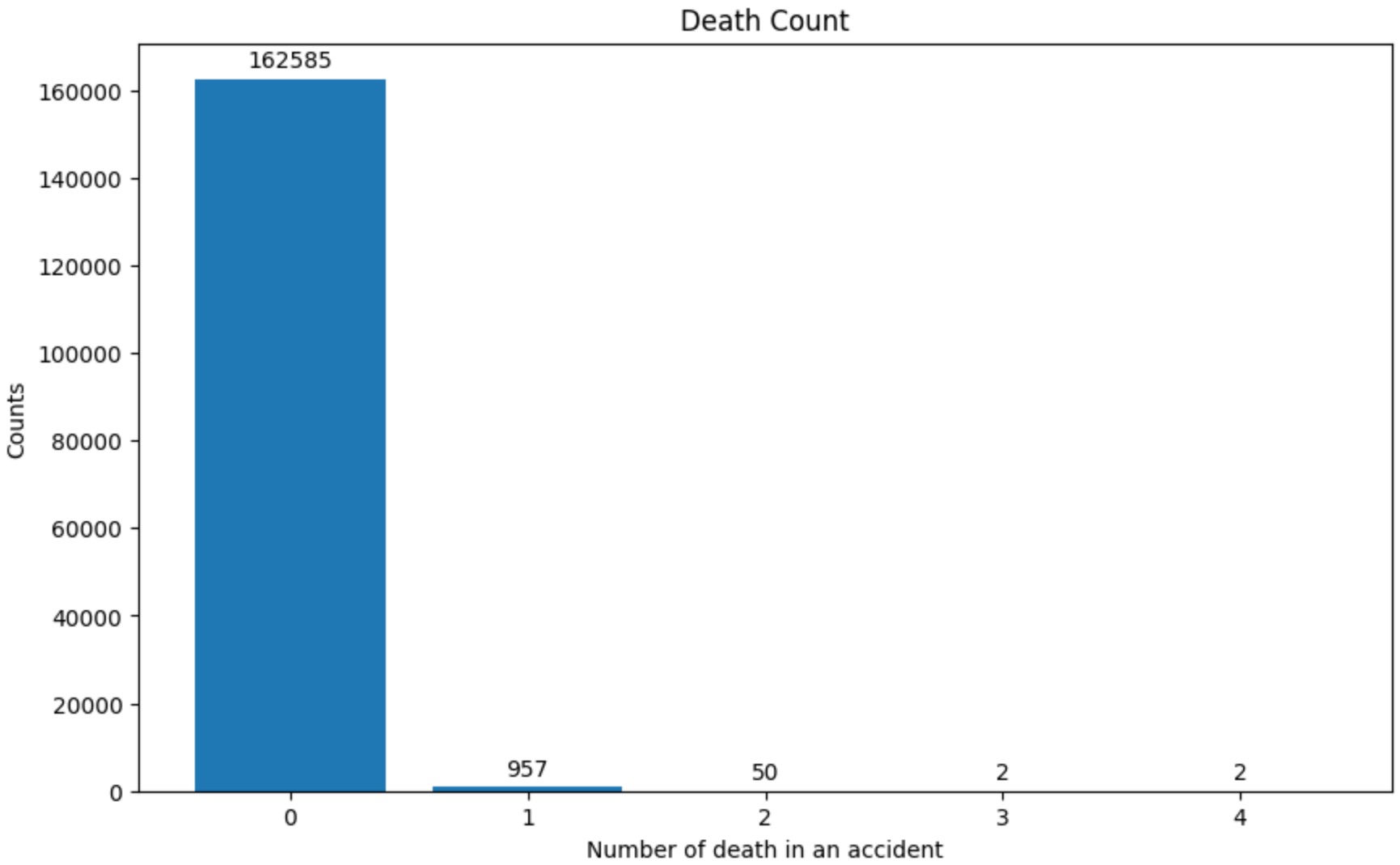
1. Removed temporary records
2. Dropped irrelevant columns such as ID, Addresses, Private/Public Roads
3. Refactored dates
4. Renamed columns for consistency
5. Value Corrections (Speed, Crash Severity, Units Involved)
6. Dropped rows with missing values.

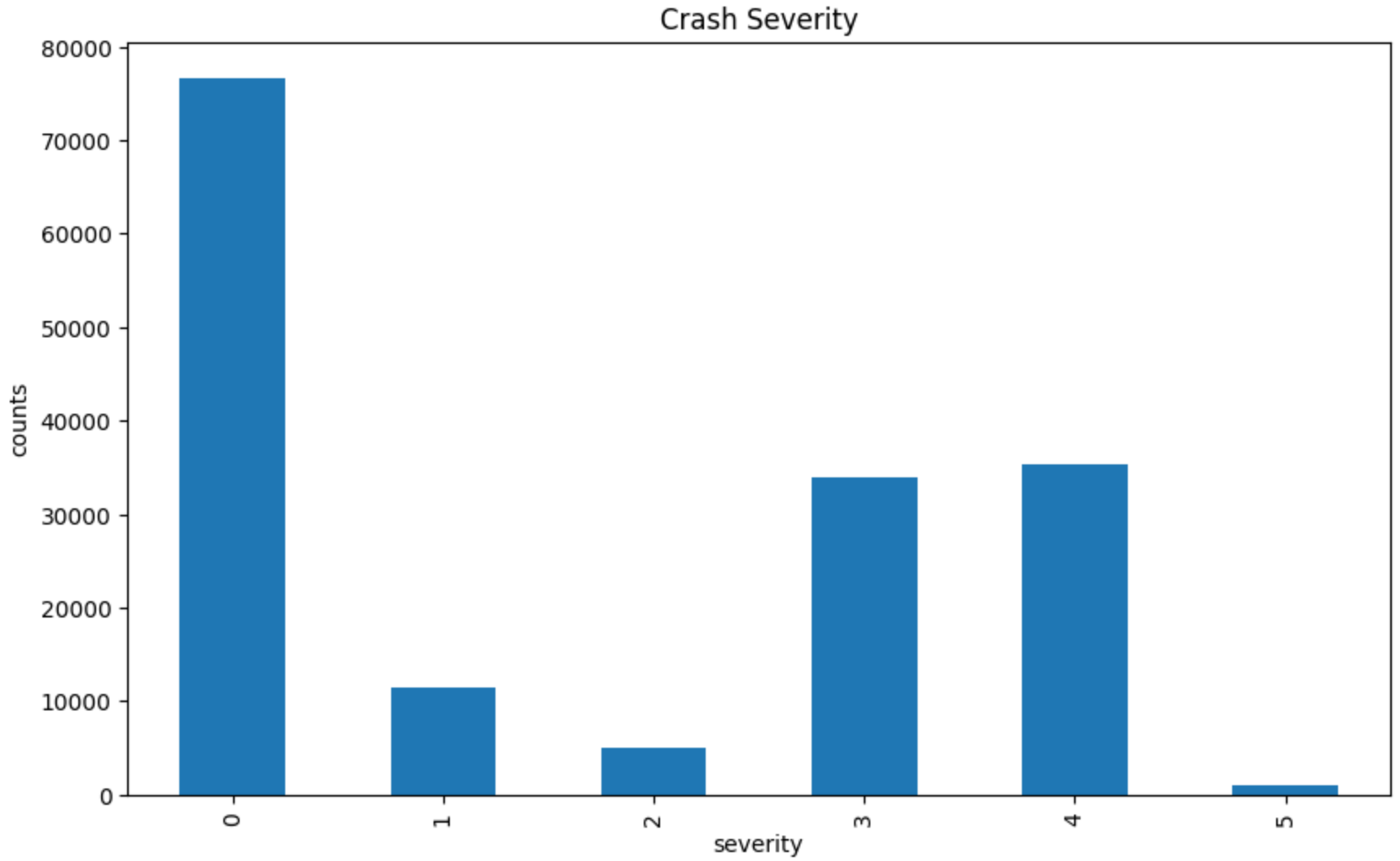


EXPLORATORY ANALYSIS

Average Total Comprehensive Cost is 307,980 $




Death Count
162585
957
50
2
2
160000
140000
120000
100000
80000
60000
40000
20000
0
Counts
0
1
2
3
4
Number of death in an accident


Crash Severity
80000
70000
60000
50000
40000
30000
20000
10000
0
counts
0
1
2
3
4
5
severity

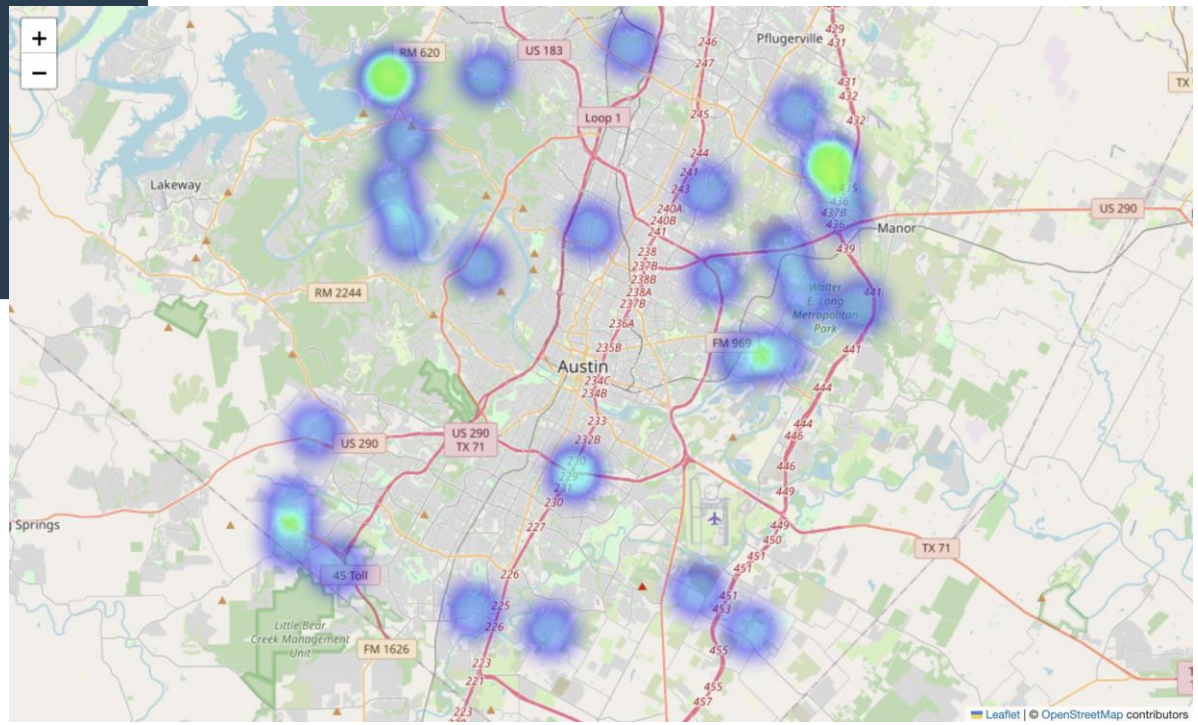
MAP

```
from folium.plugins import HeatMap

lat_avg = df['latitude'].mean()
lon_avg = df['longitude'].mean()

accidents_heat_map =
df[['latitude','longitude','death_cnt']]
lat_avg = accidents_heat_map['latitude'].mean()
lon_avg = accidents_heat_map['longitude'].mean()

map = folium.Map([lat_avg, lon_avg], zoom_start=10)
HeatMap(accidents_heat_map).add_to(map)
map
```



A better map is available at
https://visionzero.austin.gov/viewer/map

FEATURE SELECTION

ORIGINAL DATASET (AFTER CLEANSING)

- 47 features, some may be redundant,
- Risk of Overfitting and Lengthy execution time,

➔ USE BACKWARD ELIMINATION

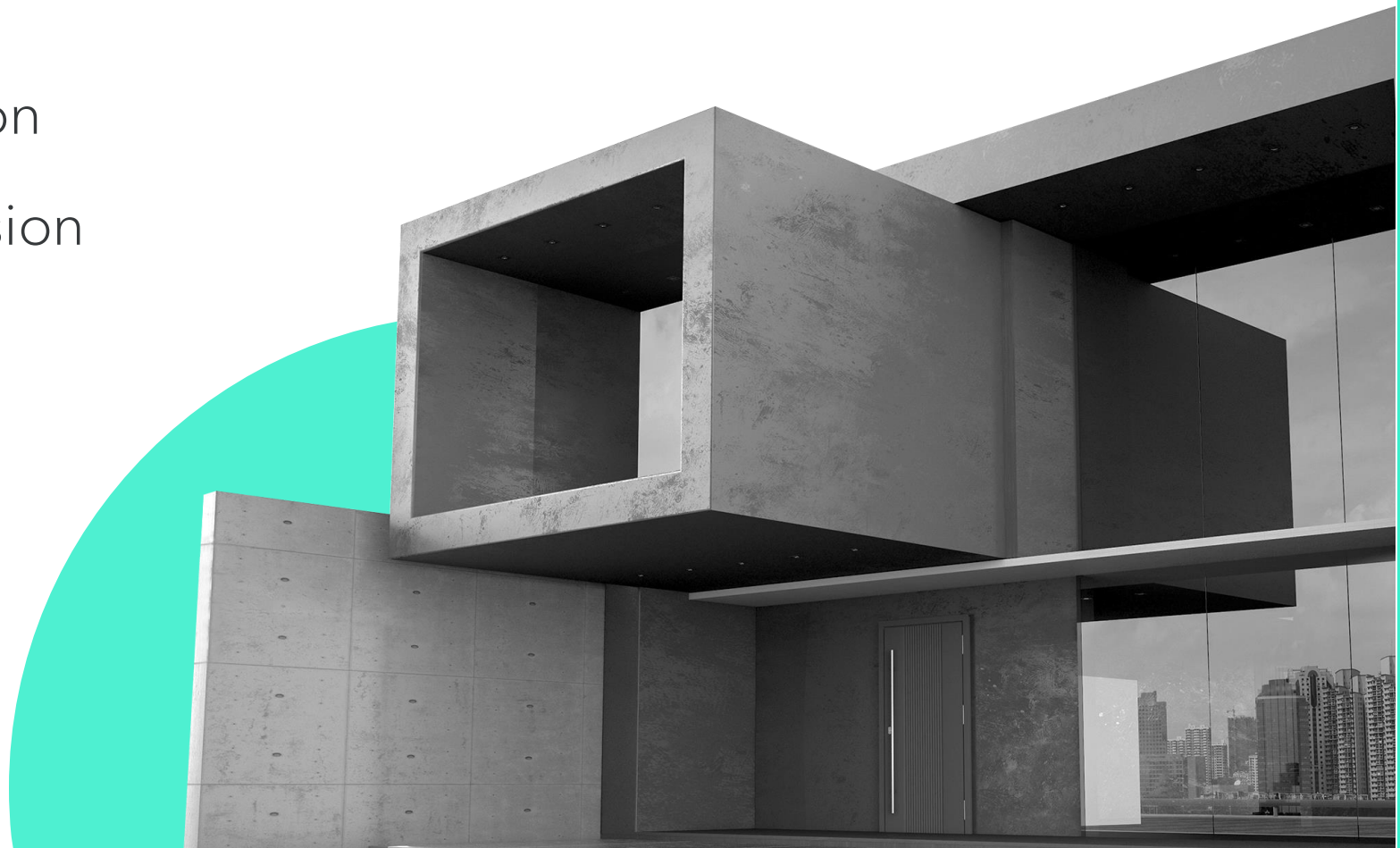
- Remove features greater than 0.05 p-value.

➔ REMAINING 27 FEATURES

- For example, Time:
  - Drop Day, Month, Year
  - Keep Hour + Weekend.

ML METHODOLOGY

- Linear Regression
- Ridge Regression
- Lasso Regression
- Decision Tree Regression
- Random Forest Regression
- SVR
- KNN



LINEAR REGRESSION

Capture linear relationship

Can explain which features impact the cost and by how much

RIDGE REGRESSION

Reduce overfitting, work well with many features

Minimize gap between training and testing model errors

LASSO REGRESSION

Leverage automatic feature selection

DECISION TREE REGRESSION

Works well with Non-Linear Relationship

RANDOM FOREST REGRESSION

Combine multiple trees to improve accuracy

Reduce overfitting

Works well for non-linear and complex relationships

SVR

Works well with high-dimensional spaces, non-linear patterns

KNN

Simple and non-parametric

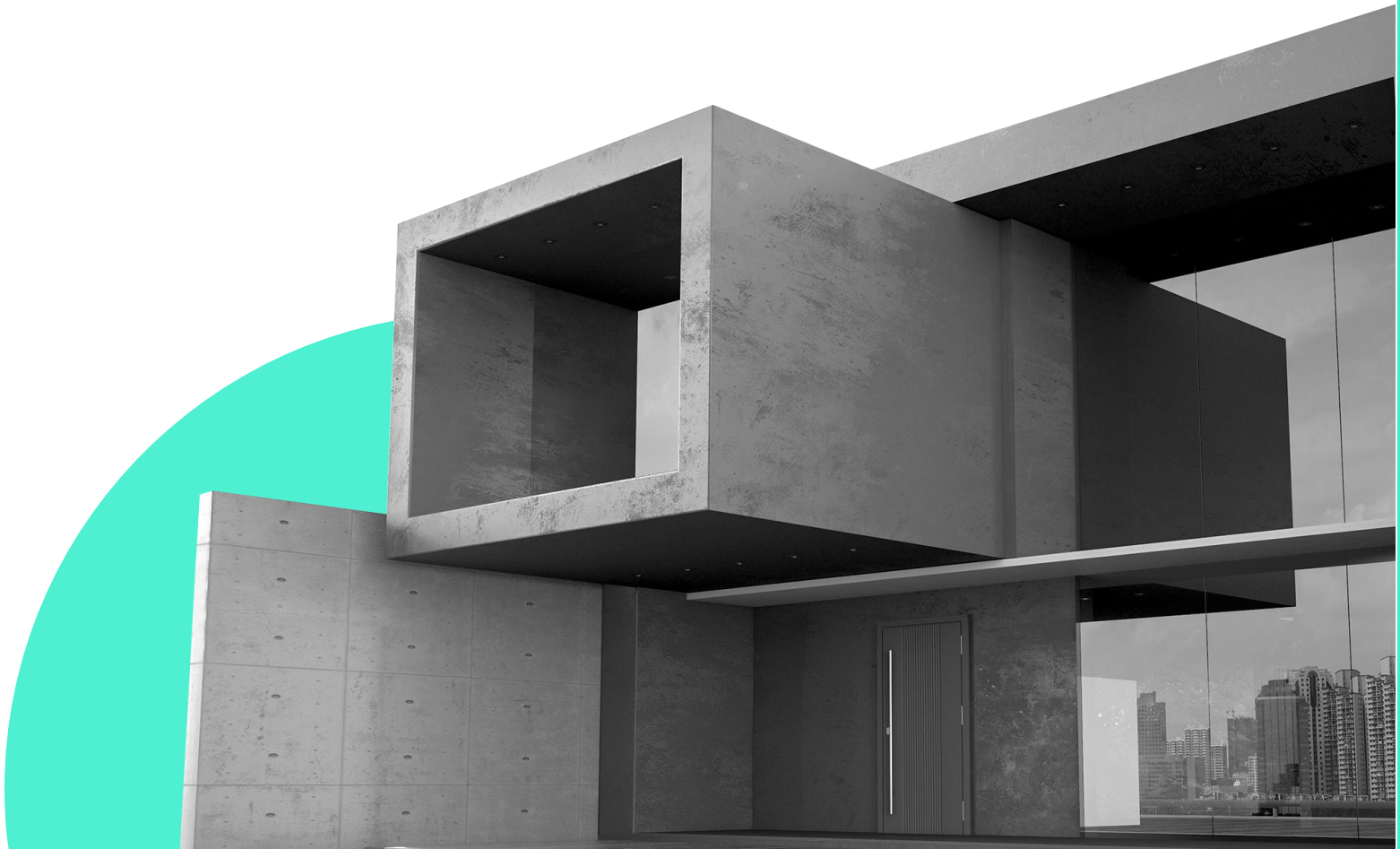
Can capture local variations in cost.

CODE

```
models = {
    'linear_regression': Pipeline([
        ('preprocessor', preprocessor),
        ('regressor', LinearRegression())
    ]),
    "ridge_regression": Pipeline([
        ('preprocessor', preprocessor),
        ('regressor', Ridge(alpha=1.0))
    ]),
    "lasso_regression": Pipeline([
        ('preprocessor', preprocessor),
        ('regressor', Lasso(alpha=0.1))
    ]),
    "decision_tree_regressor": Pipeline([
        ('preprocessor', preprocessor),
        ('regressor', DecisionTreeRegressor(random_state=RANDOM_SEED))
    ]),
    "random_forest_regressor": Pipeline([
        ('preprocessor', preprocessor),
        ('regressor', RandomForestRegressor(n_estimators=100, random_state=RANDOM_SEED, n_jobs=-1))
    ]),
    # SVR needs a better hardware
    "svr": Pipeline([
        ('preprocessor', preprocessor),
        ('regressor', SVR(kernel='rbf', C=1.0, epsilon=0.1))
    ]),
    "k_neighbors_regressor": Pipeline([
        ('preprocessor', preprocessor),
        ('regressor', KNeighborsRegressor(n_neighbors=90, n_jobs=-1))
    ])
}
```

GRIDSEARCH

We will be using the GridSearchCV method to find the best hyperparameters for our models.



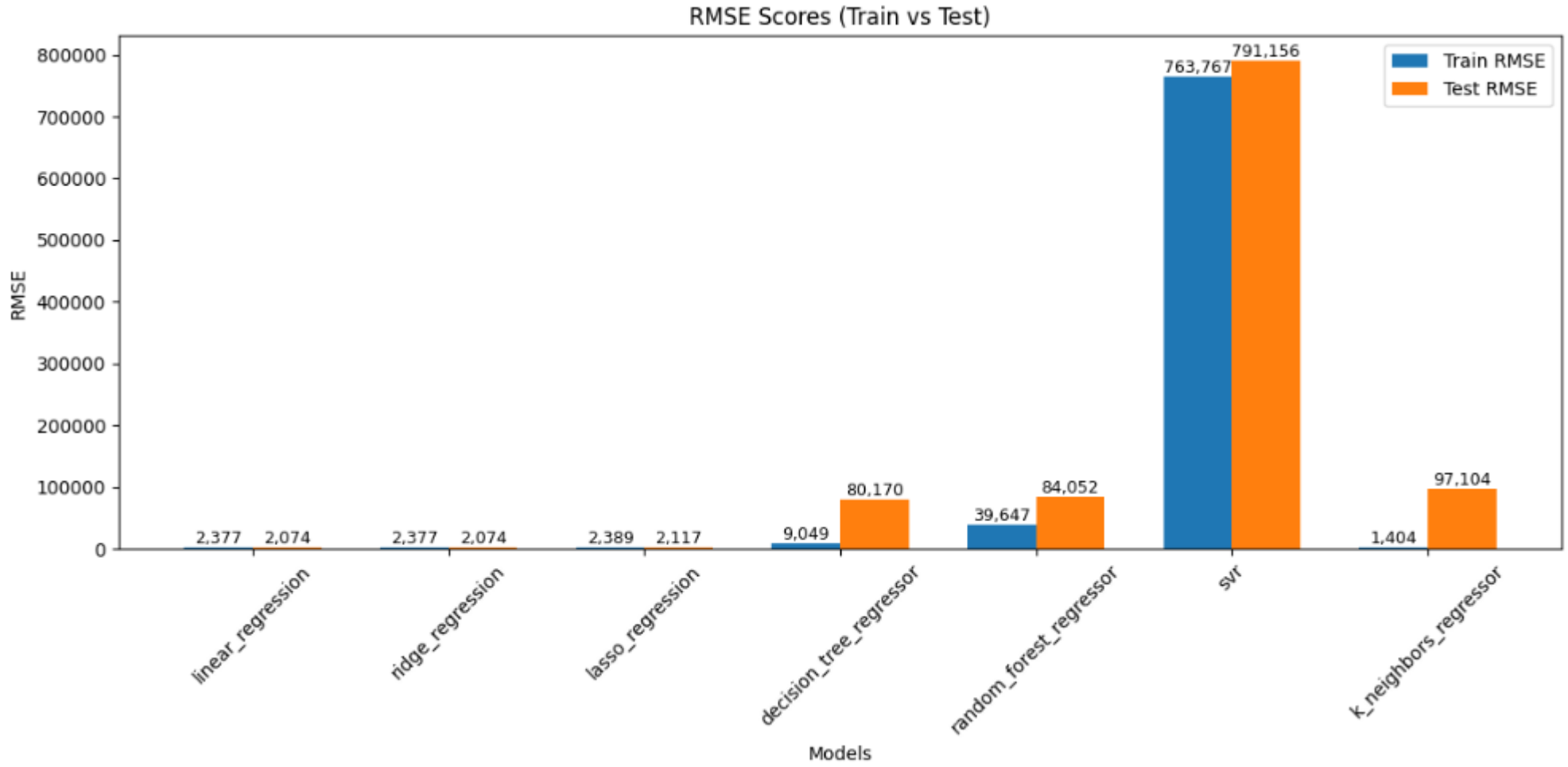
CODE

```
param_grids = {
    'linear_regression': {},
    'ridge_regression': {
        'regressor__alpha': [0.1, 1.0, 10.0]
    },
    'lasso_regression': {
        'regressor__alpha': [0.1, 1.0, 10.0]
    },
    'decision_tree_regressor': {
        'regressor__max_depth': [3,5,10],
        'regressor__min_samples_split':[2,5,10]
    },
    'random_forest_regressor': {
        'regressor__n_estimators':[50, 100, 200],
        'regressor__max_depth':[3,5,10]
    },
    'srv': {
        'regressor__C': [0.1, 1.0, 10.0], # penalty parameter
        'regressor__epsilon': [0.1, 0.2, 0.5], # no pentaly if error is within epsilon
        'regressor__kernel': ['linear', 'rbf']
    },
    'k_neighbors_regressor': {
        'regressor__n_neighbors': [3,5,10], # how many neighbors to look at
        'regressor__weights': ['uniform', 'distance'] # how to weight neighbors
    }
}
```

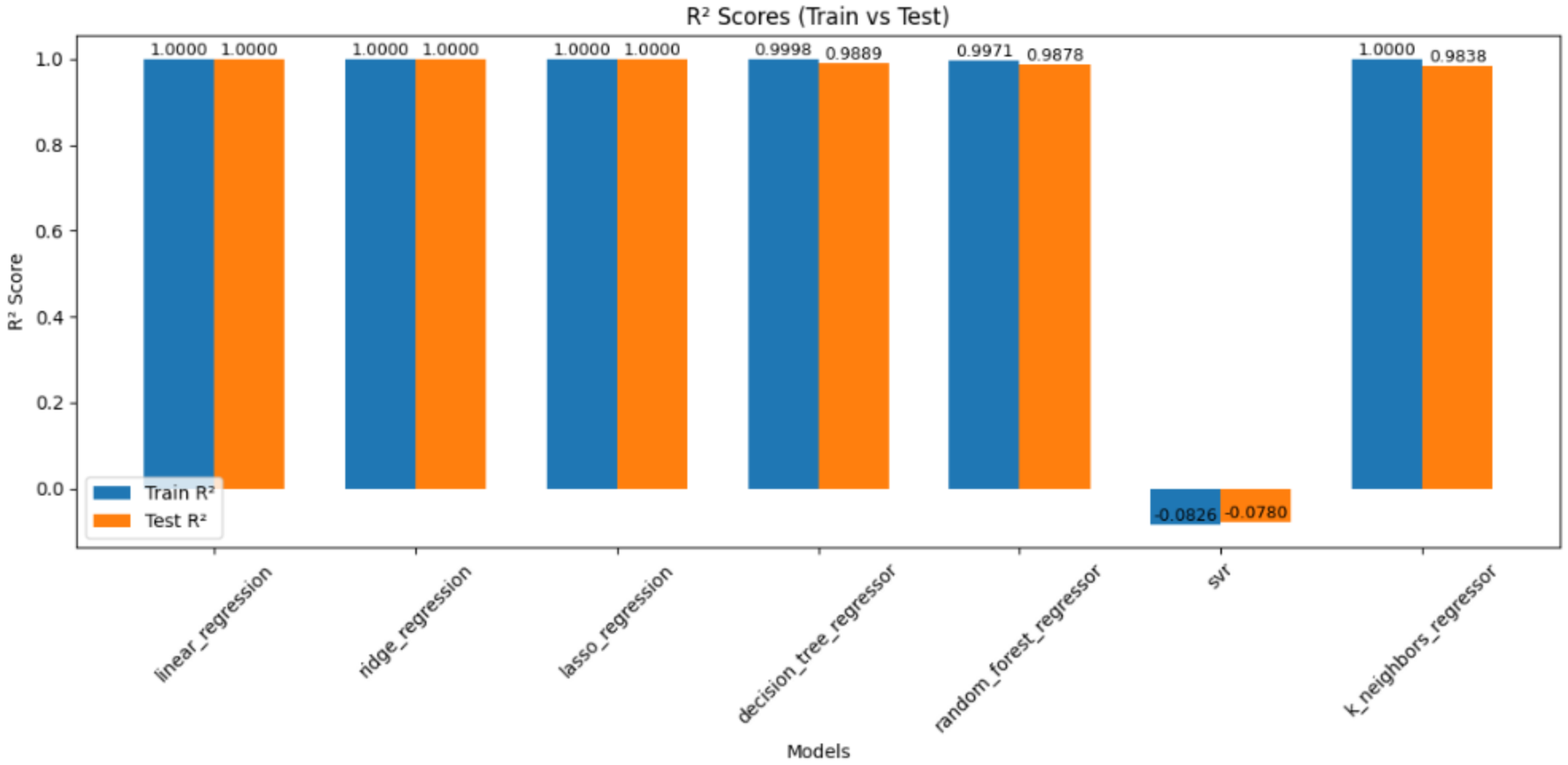
BEST HYPERPARAMETERS

- Ridge Regression: alpha 0.1
- Lasso Regression: alpha 0.1
- Decision Tree:
  - Max Depth: 10
  - Min Samples Split: 2
- Random Forest:
  - Max Depth: 10
  - Number of estimators: 100
- KNN:
  - Number of neighbors: 3
  - Weights: Distance

RMSE COMPARISON



R^2 COMPARISON



CONCLUSION

What model is best to predict comprehensive cost?

In the end, the linear model performed the best with the highest r^2 and lowest RMSE % on test set of all models. In the future, we should test more hyperparameters, models, and determine whether data leakage has occurred.

THANK YOU

HUNG TRAN, STEVEN TRAN, LUCA COMBA