QUICKVAL

YOUR PROPERTY OUR EXPERTISE

Meet Our Team

IPSHITA CHAUHAN

CHARVI NINGALA

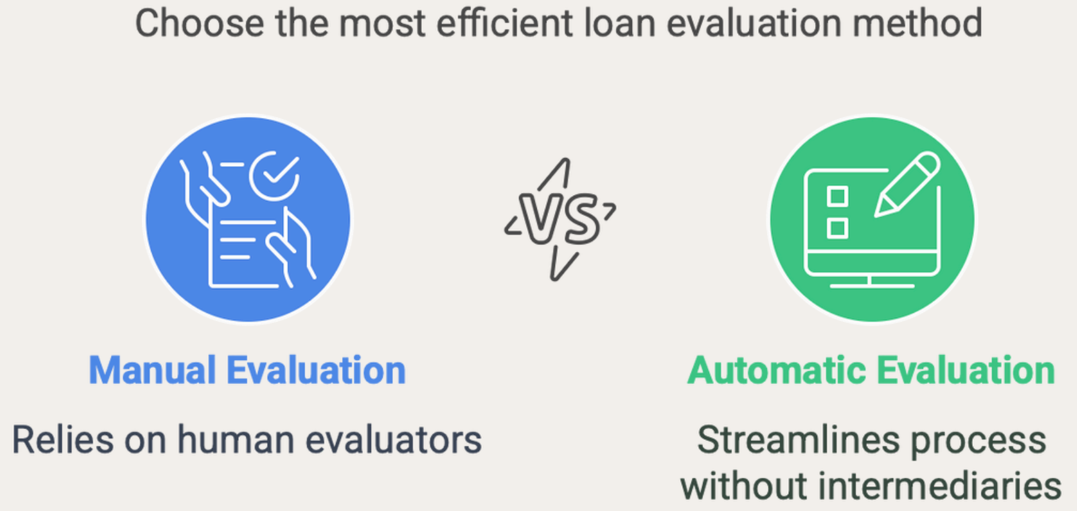
B.RAHUL REDDY

CHRIS JOSE

ARMAN SHAIK

Problem Statement

Banks today rely on the manual working of the evaluator determining the loan amount for the customer based on property. This project aims to create an automatic system that does not use a middleman like an evaluator.



Abstract

QuickVal is a system for rapidly assessing the reasonableness of a property's price. It analyzes the property age and other factors to determine if the requested price aligns with comparable market factors, providing a data-driven valuation tool for real estate transactions.

Support Vector Machine

In [14]:
```
from sklearn.model_selection import GridSearchCV

param_grid = {
    'C': [0.1, 1, 10],
    'gamma': ['scale', 0.1, 0.01, 0.001],
    'kernel': ['rbf', 'linear','sigmoid','poly']
}
```

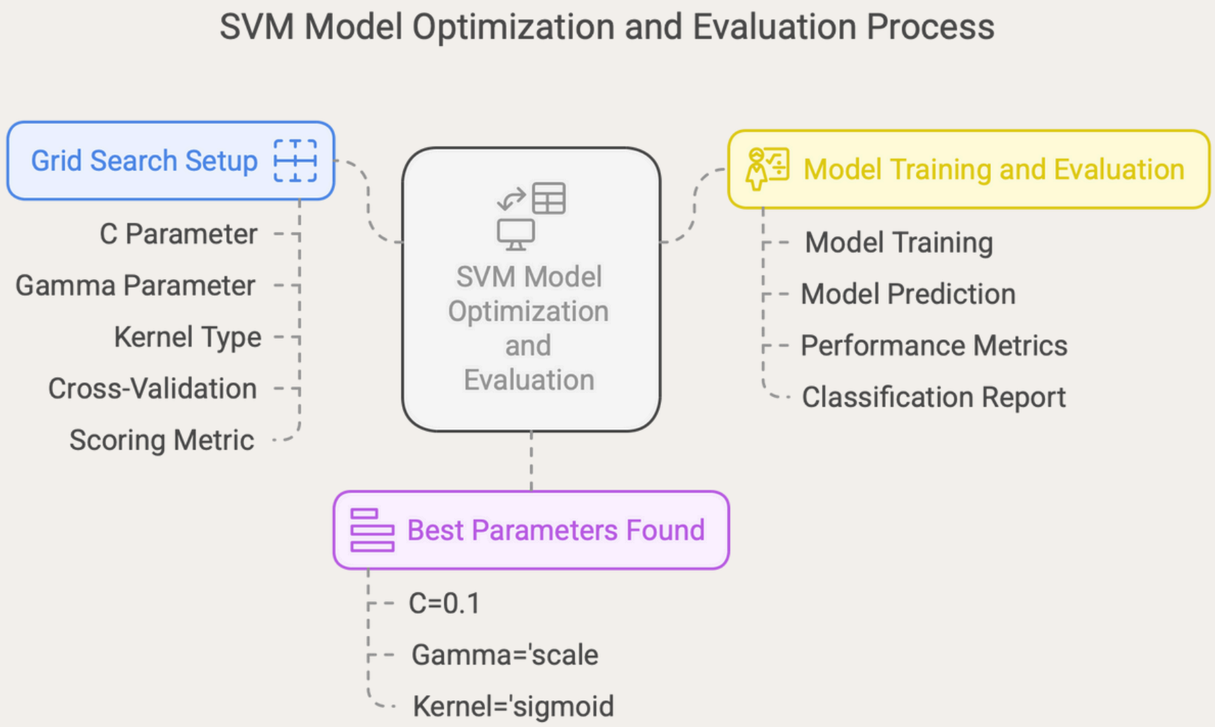
In [15]:
```
grid_search = GridSearchCV(SVC(), param_grid, cv=5, scoring='accuracy')
grid_search.fit(X_train, y_train)

print("Best Parameters:", grid_search.best_params_)
```

```
Best Parameters: {'C': 0.1, 'gamma': 'scale', 'kernel': 'sigmoid'}
```

In [16]:
```
svm_best = SVC(**grid_search.best_params_)
svm_best.fit(X_train, y_train)
y_pred = svm_model.predict(X_test)
```

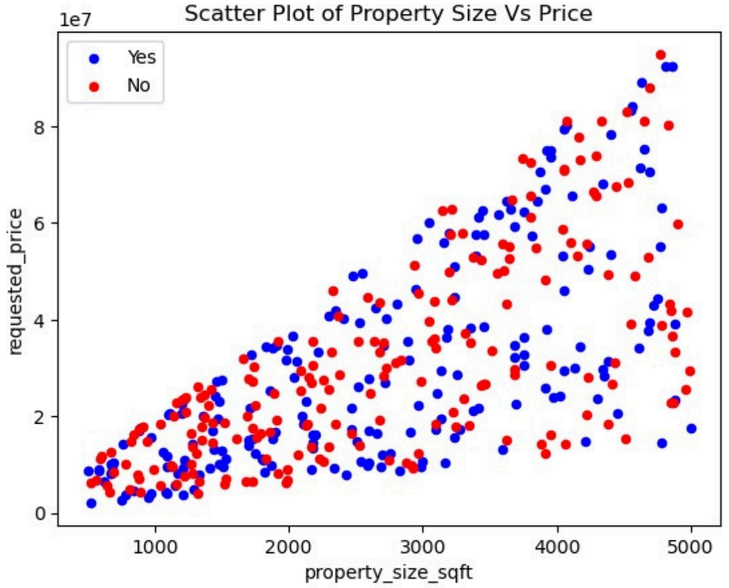
In [23]:
```
y_pred = svm_model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)

print(f"SVM Model Accuracy: {accuracy :.2f}")
print(classification_report(y_test, y_pred))
```

```
SVM Model Accuracy: 0.70
              precision    recall  f1-score   support

           0       0.72      0.63      0.67      1205
           1       0.69      0.77      0.73      1295

    accuracy                           0.70      2500
   macro avg       0.70      0.70      0.70      2500
weighted avg       0.70      0.70      0.70      2500
```



Scatter Plot

```
In [7]: data.isna().any().any()
Out[7]: False

In [8]: cl1=data[data['approval_status']==1][0:200]
        cl2=data[data['approval_status']==0][0:200]
        a=cl1.plot(kind='scatter',x='property_size_sqft',y='requested_price',color='blue',label='Yes')
        cl2.plot(kind='scatter',x='property_size_sqft',y='requested_price',color='red',label='No',ax=a)
        plt.title("Scatter Plot of Property Size Vs Price")
        plt.legend()
        plt.show()
```



Explanation

1. Data Inspection:
   - Missing Values Check: The code checks for missing values in the dataset using data.isna().any().any().
   - Result: No missing values are found (Out 7: False).
2. Data Selection and Visualization:
   - Data Selection:
     - Selects the first 200 rows where approval_status is 1 (cl1) and 0 (cl2).
   - Scatter Plot:
     - Creates a scatter plot to visualize the relationship between property_size_sqftand requested_price.
     - cl1 (approved) is plotted in blue, labeled as "Yes".
     - cl2 (not approved) is plotted in red, labeled as "No".
   - Plot Details:
     - The plot shows how requested price varies with property size for both approved and not approved statuses.
     - The title of the plot is "Scatter Plot of Property Size Vs Price".

Neural Network

In [17]:
```
from tensorflow.keras.layers import Dropout,Input
from sklearn.utils.class_weight import compute_class_weight
class_weights = compute_class_weight(class_weight='balanced',
                                     classes=np.unique(y_train.numpy()),
                                     y=y_train.numpy().astype(int))

class_weight_dict = {i: class_weights[i] for i in range(len(class_weights))}

model = Sequential([
    Dense(32, activation='relu', input_shape=(X_train.shape[1],)),
    Dropout(0.3),  # Drop 30% of neurons randomly
    Dense(16, activation='relu'),
    Dropout(0.3),
    Dense(1, activation='sigmoid')
])
```

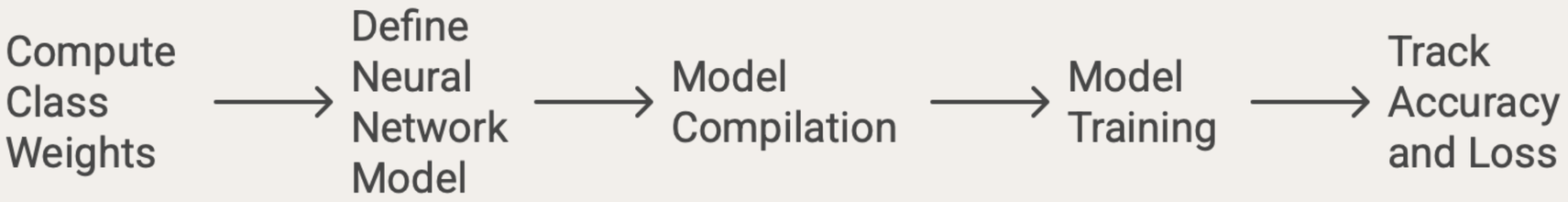
```
C:\Users\User\anaconda3\lib\site-packages\keras\src\layers\core\dense.py:87: UserWarning: Do not pass an `input_shape`/`inpu
t_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the mo
del instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

In [18]:
```
from tensorflow.keras.optimizers import Adam

optimizer = Adam(learning_rate=0.001)  # Try reducing to 0.0005 or lower if needed
model.compile(optimizer=optimizer, loss='binary_crossentropy', metrics=['accuracy'])
```

Neural Network

```
In [19]: history = model.fit(X_train, y_train, epochs=50, batch_size=32, validation_data=(X_test, y_test),class_weight=class_weight_d

Epoch 1/50
79/79 ━━━━━━━━━━━━━━━━━━━━ 3s 11ms/step - accuracy: 0.5180 - loss: 0.7838 - val_accuracy: 0.5628 - val_loss: 0.6821
Epoch 2/50
79/79 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - accuracy: 0.5367 - loss: 0.7186 - val_accuracy: 0.5968 - val_loss: 0.6711
```

Neural Network Model Development Process



Neural Network

```
[21]: loss, accuracy = model.evaluate(X_test, y_test)
      print(f"Test Accuracy: {accuracy:.2f}")
```

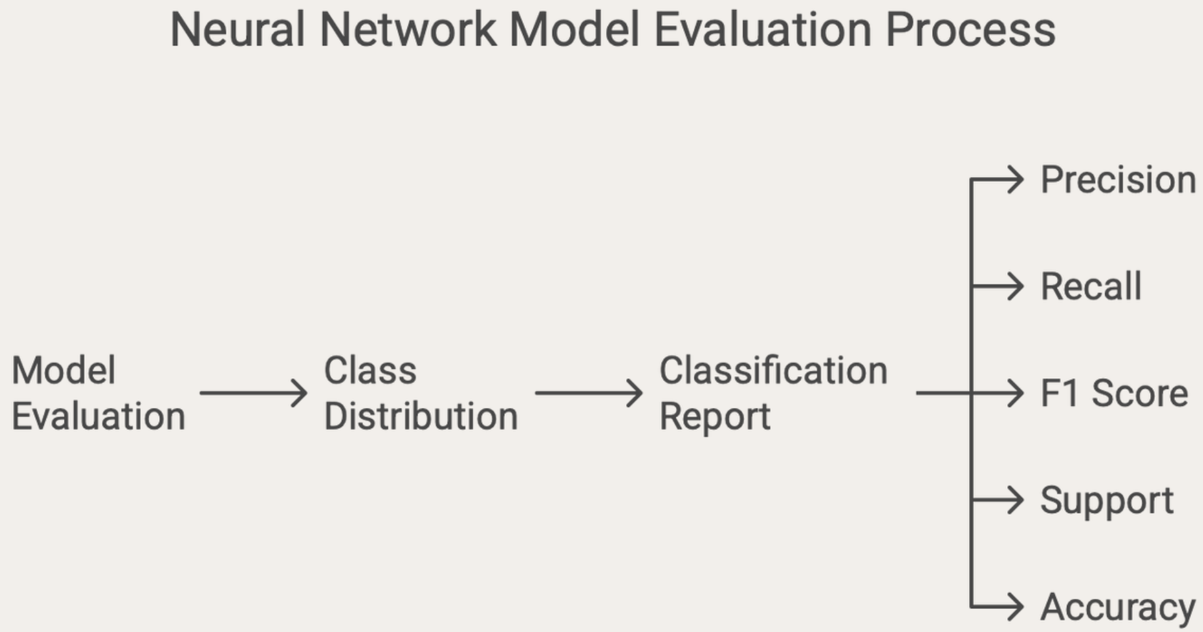
```
79/79 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7625 - loss: 0.4980
Test Accuracy: 0.75
```

```
[22]: y_train_np = y_train.numpy()
      print("Class distribution:", np.bincount(y_train_np.astype(int)))
```

```
Class distribution: [1150 1349]
```

```
[23]: from sklearn.metrics import classification_report

      y_pred = model.predict(X_test)
      y_pred = (y_pred > 0.5).astype(int)

      print(classification_report(y_test, y_pred))
```

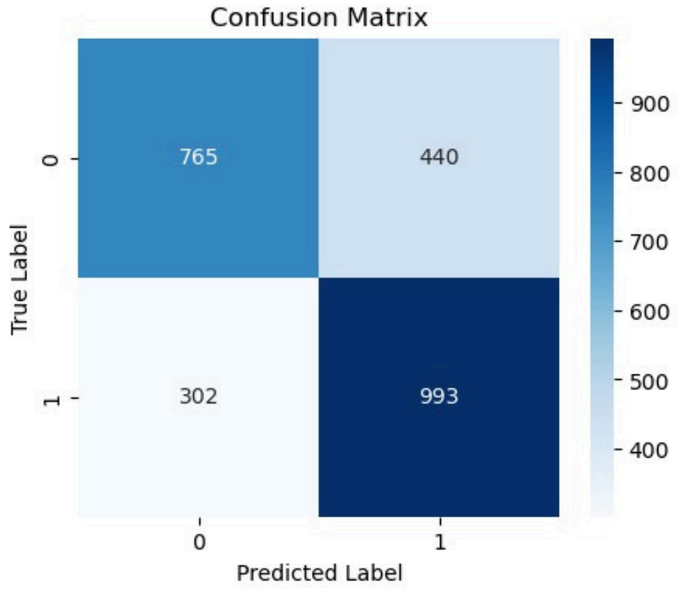
```
79/79 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
              precision    recall  f1-score   support

         0.0       0.87      0.57      0.69      1205
         1.0       0.70      0.92      0.79      1295

    accuracy                           0.75      2500
   macro avg       0.78      0.74      0.74      2500
weighted avg       0.78      0.75      0.74      2500
```

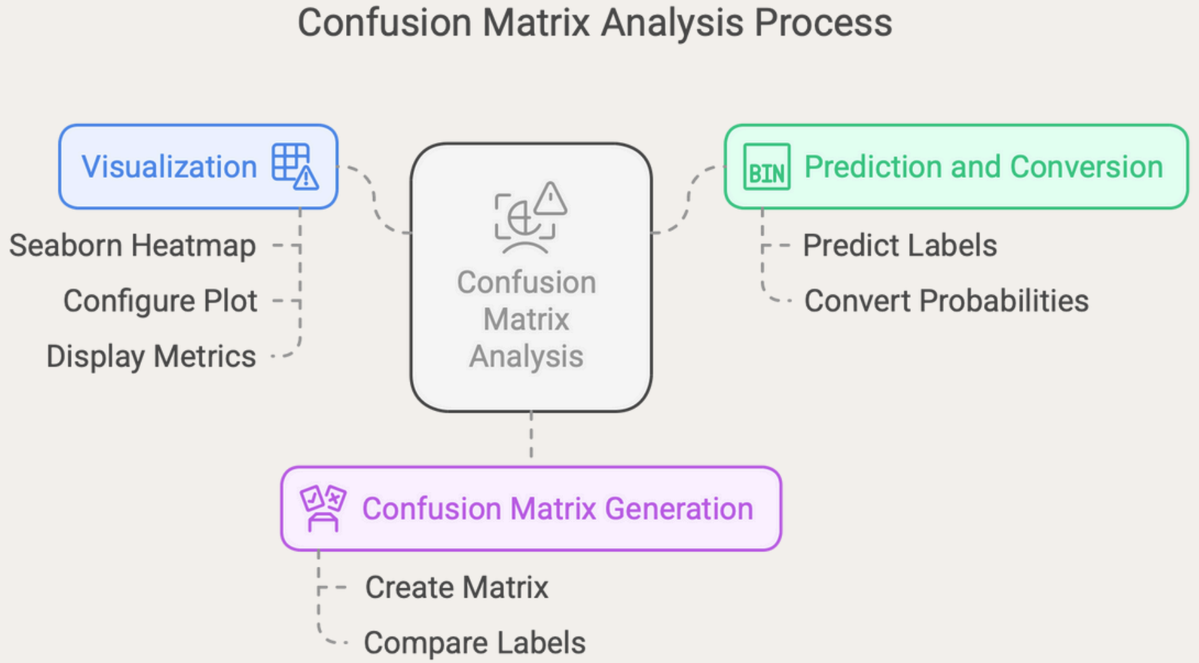


Confusion Matrix

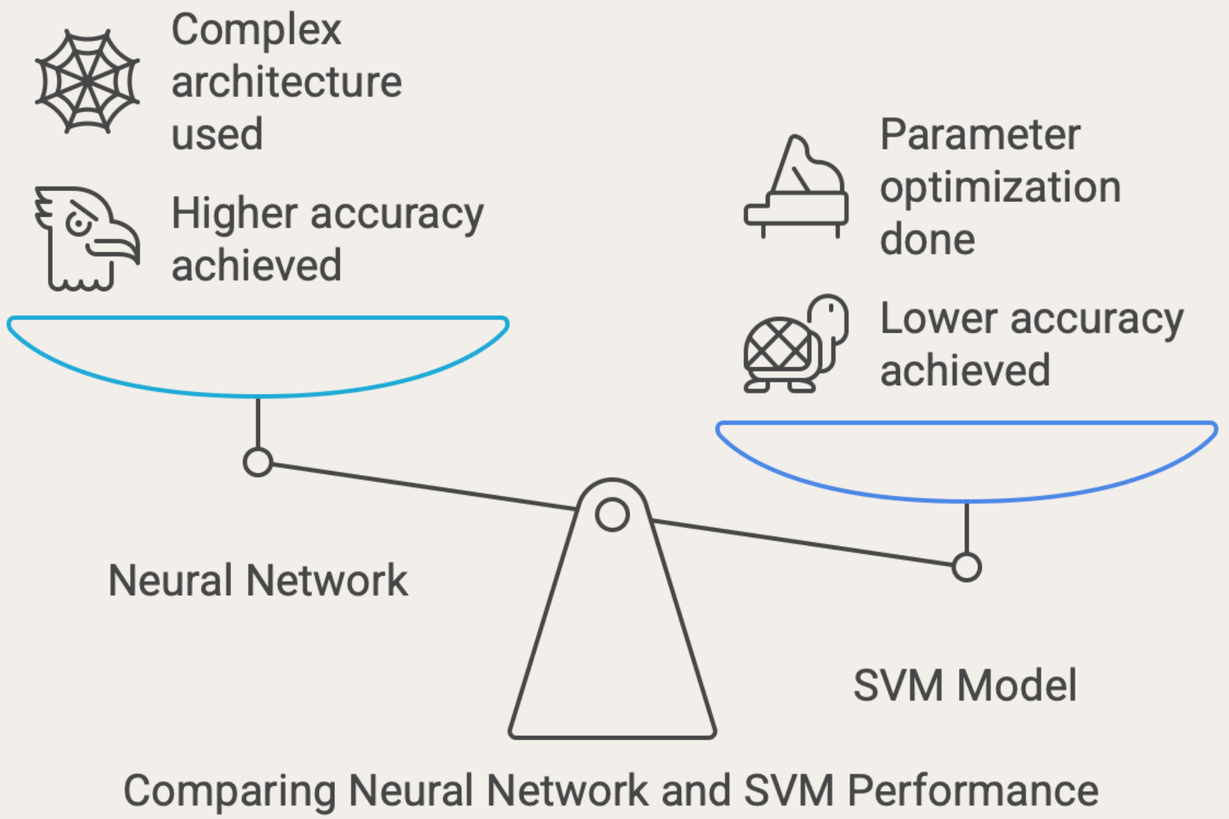
19]:

```
from sklearn.metrics import confusion_matrix
import seaborn as sns

y_pred = (svm_model.predict(X_test) >= 0.5).astype(int)
cm = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(5, 4))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("Confusion Matrix")
plt.show()
```





SVM v/s NN



Comparing Neural Network and SVM Performance

Future Plan

Basic Dashboard

Offers a simple interface with fast development using React.js or HTML/CSS/JS.

Comparative Pricing Graph

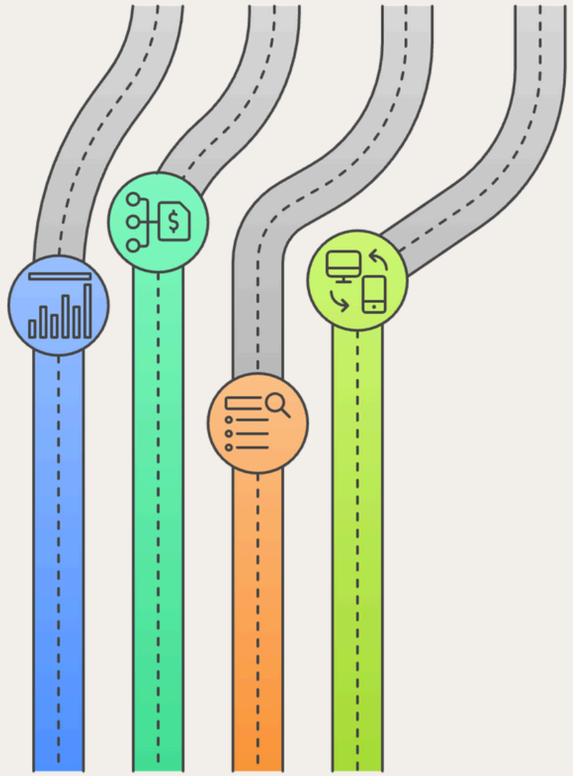
Provides visual justification of property prices using Chart.js or Recharts.

Search & Filtering

Enhances usability with search and filters using JavaScript or React hooks.

Mobile-Friendly UI

Ensures accessibility on desktop and mobile using Tailwind CSS or Bootstrap.



THANK YOU