SUPERIOR UNIVERSITY

Name :	**MUHAMMAD AHMAD**
Roll No :	**SU92-BSAIM-F23-135**
Section :	**4-C**
Task :	**01**
Date :	**05th March, 2025**
Subject :	**Programming AI Lab**
Submitted To :	**Prof Rasikh Ali**

TASK : 01

House Price Prediction

1. Introduction

This project involves predicting house prices using Support Vector Regression (SVR). The dataset consists of training data (train_home.csv), test data (test_home.csv), and a sample submission file (sample_submission_home.csv). The preprocessing steps include handling missing values, scaling numerical features, and encoding categorical features before training an SVR model.

2. Data Loading and Copying

```
import pandas as pd

train_data = pd.read_csv('train_home.csv')

test_X = pd.read_csv('test_home.csv')

test_y = pd.read_csv('sample_submission_home.csv')
```

- train_home.csv: Contains house features and target variable (SalePrice).
- test_home.csv: Contains house features without SalePrice (for prediction).
- sample_submission_home.csv: Example format for submission.

We make copies of the datasets to ensure the original data remains unchanged:

```
encoded_train = train_data.copy()

encoded_test_X = test_X.copy()

encoded_test_y = test_y.copy()
```

3. Feature Selection and Removal

```
def delete_col(df, col):
    for i in col:
        df.drop(i, axis=1, inplace=True)


col = ['Id', 'Alley', 'Street', 'MasVnrType', 'MiscFeature', 'Fence', 'PoolQC', 'FireplaceQu', 'MoSold']

delete_col(encoded_train, col)

delete_col(encoded_test_X, col)
```

- Drops columns with too many missing values or low predictive power.

4. Handling Missing Values

For Numerical Columns (Outlier Handling & Filling Nulls)

```
def fill_null(df, col):
    for i in col:
        q1 = df[i].quantile(0.25)
        q3 = df[i].quantile(0.75)
        iqr = q3 - q1
        lower_lim = q1 - 1.5 * iqr
        upper_lim = q3 + 1.5 * iqr
        without_outlier = df[i].apply(lambda x: None if x < lower_lim or x > upper_lim else x)
        without_outlier.fillna(without_outlier.mean(), inplace=True)
        df[i] = without_outlier
```

- Identifies outliers using the interquartile range (IQR) method and replaces them with the column mean.

For Categorical Columns (Filling with Mode)

```
def fill_obj(df, col):
    for i in col:
        df[i].fillna(df[i].mode()[0], inplace=True)
```

- Fills missing categorical values with the most frequent category (mode).

5. Identifying Numerical and Categorical Columns

```
def int_col(df):
    return [i for i in df.columns if df[i].dtype != 'O']


def obj_col(df):
    return [i for i in df.columns if df[i].dtype == 'O']
```

- int_col(df): Returns a list of numerical columns.

- obj_col(df): Returns a list of categorical columns.

6. Feature Scaling

Standard Scaling for Continuous Features

```
from sklearn.preprocessing import StandardScaler

std = StandardScaler()

std_list = ['LotFrontage', 'LotArea', 'MasVnrArea', 'BsmtFinSF1', 'BsmtFinSF2', 'TotalBsmtSF',
        '1stFlrSF', '2ndFlrSF', 'GrLivArea', 'GarageArea']

encoded_train[std_list] = std.fit_transform(encoded_train[std_list])

encoded_test_X[std_list] = std.transform(encoded_test_X[std_list])
```

- StandardScaler transforms features to have zero mean and unit variance.
- Fit is applied on training data, and transform is applied on both training and test sets.

Min-Max Scaling for Other Numerical Features

```
from sklearn.preprocessing import MinMaxScaler

minmax = MinMaxScaler()

minmax_list = ['YearBuilt', 'YearRemodAdd', 'GarageYrBlt', 'YrSold', 'BedroomAbvGr', 'TotRmsAbvGrd',
          'Fireplaces', 'GarageCars', 'FullBath', 'HalfBath', 'BsmtFullBath', 'BsmtHalfBath']

encoded_train[minmax_list] = minmax.fit_transform(encoded_train[minmax_list])

encoded_test_X[minmax_list] = minmax.transform(encoded_test_X[minmax_list])
```

- MinMaxScaler transforms features to a range between 0 and 1.

7. Encoding Categorical Features

```
from sklearn.preprocessing import LabelEncoder


def obj_to_int(df, col):

    for i in col:

        label = LabelEncoder()

        df[i] = label.fit_transform(df[i])
```

- Converts categorical features into numerical values using LabelEncoder.

8. Splitting Features and Target Variable

train_X = encoded_train.drop('SalePrice', axis=1)

train_y = encoded_train['SalePrice']

- train_X: Features for training.
- train_y: Target variable (house prices).

9. Model Training (Support Vector Regression - SVR)

from sklearn.svm import SVR

svr_model = SVR()

svr_model.fit(train_X, train_y)

- Uses SVR, a regression algorithm based on Support Vector Machines (SVMs).
- The default kernel is rbf (Radial Basis Function).

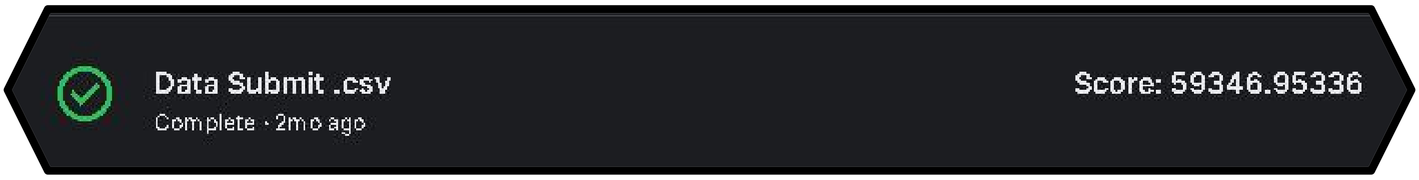
10. Making Predictions

pred = svr_model.predict(encoded_test_X)

- Predicts house prices using the trained SVR model.

11. Conclusion

This project preprocesses house price data handles missing values, scales numerical features, encodes categorical features, and applies an SVR model for price prediction. The next step is to evaluate the model's performance using metrics such as RMSE (Root Mean Squared Error) or R^2 Score. Further improvements can be made by experimenting with different models like Random Forest or XGBoost.



Data Submit .csv
Complete · 2mo ago
Score: 59346.95336