DS605 Fundamentals of Machine Learning
Academic Year 2025-26 (Autumn)

LAB - 4

Data Preprocessing and Regression With Scikit-learn

Objective: Your goal is to build a robust, end to end machine learning pipeline to predict house prices for the King County dataset. This assignment requires you to move beyond the manual steps in the original notebook and implement a more sophisticated, automated preprocessing workflow using Scikit Learn's Pipeline and ColumnTransformer. You will also perform advanced feature engineering and hyperparameter tuning to maximize model performance.

Task 1: Advanced Exploratory Data Analysis (EDA) & Feature Engineering

Before building the model, you must deeply understand and enrich the dataset.

Introduce and Handle Missing Data: The original dataset is clean. Real world data is not. To simulate this, randomly introduce approximately 5% missing values into the sqft_lot and bathrooms columns. You'll need to devise an imputation strategy for these later in your pipeline.

1. Outlier Detection and Handling:
    - Create boxplots for price, sqft_living, and bedrooms.
    - Identify and remove egregious outliers. A common rule is to remove data points that lie beyond 1.5timesIQR (Interquartile Range) from the first or third quartile. Justify your outlier removal strategy. *For instance, does a house with 33 bedrooms seem plausible?*
2. Advanced Feature Engineering: Create the following new features, as they often provide more predictive power than raw data:
    - Date-Based Features: Do not drop the date column. Convert it to a datetime object and engineer the following:

        - sale_year: The year the house was sold.
        - sale_month: The month the house was sold.
        - house_age: The age of the house at the time of sale (sale_year - yr_built).
    - Renovation Status:
        - was_renovated: A binary feature (1 if yr_renovated is not 0, else 0).
        - age_since_renovation: Years since renovation. If never renovated, this could be the same as house_age. Think about the best way to handle this.
    - Ratio Features: Create at least two insightful ratio features. For example:
        - sqft_living_per_floor: sqft_living / floors. (Handle cases where floors might be 0).
        - bath_per_bed: bathrooms / bedrooms. (Handle cases where bedrooms might be 0).

Task 2: Build a Sophisticated Preprocessing Pipeline

This is the core of the assignment. Instead of manually applying transformations, you must automate the process using sklearn.pipeline.Pipeline and sklearn.compose.ColumnTransformer. This approach is less error prone and is standard practice in production environments.

1. Identify Feature Types: After your feature engineering in Task 1, categorize your final columns into three groups:
    - Numerical Features: Continuous variables that need imputation and scaling (e.g., sqft_living, house_age, sqft_lot).
    - Categorical Features: Features that should be treated as categories, not numbers (e.g., zipcode).
    - Passthrough Features: Features you believe don't need scaling or encoding but are still useful (e.g., grade, condition, was_renovated).
2. Create Preprocessing Pipelines for Each Type:

- Numerical Pipeline: Create a Pipeline that first imputes missing values (using SimpleImputer with a median strategy) and then scales the data (using StandardScaler).
- Categorical Pipeline: Create a Pipeline that applies OneHotEncoder to the zipcode column. Set handle_unknown='ignore' to prevent errors if the test set contains a zipcode not seen in the training set.

3. Combine Pipelines with ColumnTransformer:
    - Use ColumnTransformer to apply your numerical pipeline to the numerical columns, your categorical pipeline to the categorical columns, and specify 'passthrough' for the passthrough columns.
    - This ColumnTransformer will be the first step in your final modeling pipeline.

Task 3: Model Training and Hyperparameter Tuning

Now, find the best possible model by tuning its hyperparameters.

1. Select Candidate Models: Based on the results from the original notebook, choose the top 3 performing models (e.g., Gradient Boosting, XGBoost, Random Forest).

Create a Final Pipeline: For each of the three models, create a main Pipeline that chains your ColumnTransformer (from Task 2) and the model estimator. For example:
Python

```
from sklearn.pipeline import Pipeline
from sklearn.ensemble import RandomForestRegressor

# preprocessor is your ColumnTransformer from Task 2
final_pipeline = Pipeline(steps=[('preprocessor', preprocessor),
                                 ('regressor', RandomForestRegressor(random_state=42))])
```

2. Hyperparameter Tuning with GridSearchCV:
    - For each of your three final pipelines, define a parameter grid (param_grid) to search. For the regressor step, remember to prefix the parameter names with regressor__ (e.g., 'regressor__n_estimators': [100, 200]).
    - Choose at least two parameters to tune for each model.
    - Use GridSearchCV with 3 fold cross-validation (cv=3) to find the best set of hyperparameters for each model.
    - Train GridSearchCV on the full training data. It will automatically handle applying the preprocessing pipeline correctly during cross-validation.

Task 4: Evaluation and Interpretation

Finally, evaluate your best model and interpret its results.

1. Final Evaluation:
    - Identify the best overall model and its parameters from your GridSearchCV results.
    - Evaluate this best tuned pipeline on the test set.
    - Report the final R2 score, Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE).
2. Performance Comparison:
    - Create a table or bar chart comparing the RMSE of your final, tuned model to the RMSE of the best model from the *original notebook*.
    - Calculate and report the percentage improvement in RMSE.
3. Feature Importance:
    - If your best model was a tree based model (like RandomForest or XGBoost), extract and plot the feature importances.
    - Important: The feature names from the OneHotEncoder will be generic. You'll need to retrieve the original categorical feature names to make your plot interpretable.

- Write a brief summary analyzing the top 5 most important features. Do they make sense? Are any of your engineered features in the top 5?