CS50 – FINAL PROJECT

Kaggle Competition

House Prices: Advanced Regression Techniques

Intro

- **Project Name:**
  - Kaggle Competition House Prices: Advanced Regression Techniques
- **Name**: Omar Mohamed Elbasha
- **GitHub**: Omar-Elbasha
- **EDX**: OmarElbasha
- **City**: Cairo, Egypt – currently in Dubai, United Arab Emirates
- **Date**: 21/11/2024

House Prices - Advanced Regression Techniques

Submit Prediction

Overview | Data | Code | Models | Discussion | Leaderboard | Rules | Team | Submissions

Competition Description

Ask a home buyer to describe their dream house, and they probably won't begin with the height of the basement ceiling or the proximity to an east-west railroad. But this playground competition's dataset proves that much more influences price negotiations than the number of bedrooms or a white-picket fence.

With 79 explanatory variables describing (almost) every aspect of residential homes in Ames, Iowa, this competition challenges you to predict the final price of each home.

Practice Skills

- Creative feature engineering
- Advanced regression techniques like random forest and gradient boosting

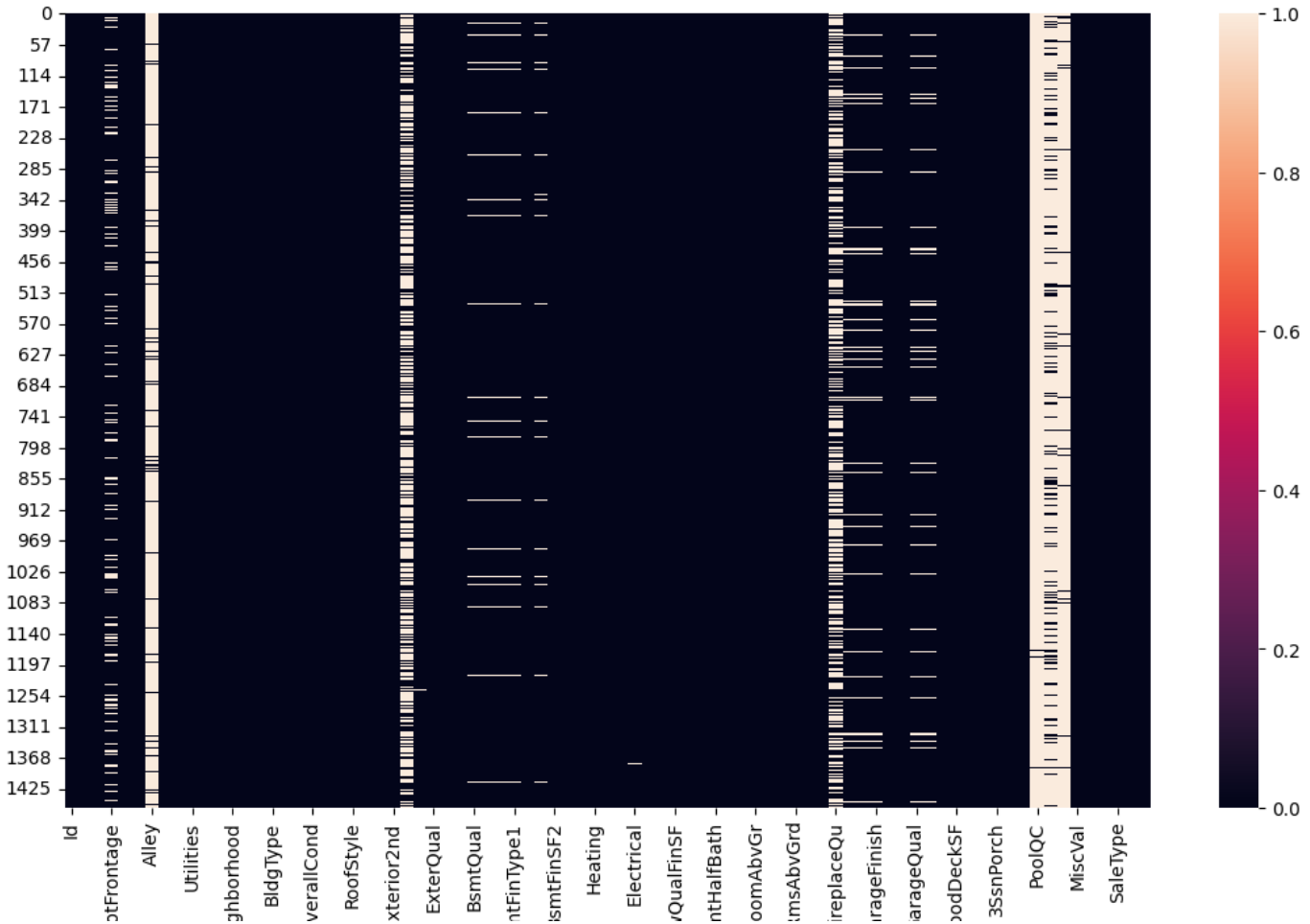
Acknowledgments

The Ames Housing dataset was compiled by Dean De Cock for use in data science education. It's an incredible alternative for data scientists looking for a modernized and expanded version of the often cited Boston Housing dataset.

Photo by Tom Thain on Unsplash.

Table of Contents

Missing Values



Feature Engineering

```python
def modify_features(df):

    df = df.drop(columns=[
        'PoolQC', 'MiscFeature', 'Alley',
        'GarageYrBlt', 'GarageCond','BsmtFinType2','Fence'])

    df['SalePrice'] = np.log1p(df['SalePrice'])
    df['HouseAge'] = df['YrSold'] - df['YearBuilt']
    df['Remodeled'] = np.where(df['YearBuilt'] == df['YearRemodAdd'], 0, 1)
    df['TotalBathrooms'] = df['FullBath'] + (0.5 * df['HalfBath']) + df['BsmtFullBath'] + (0.5 * df['BsmtHalfBath'])
    df = df.drop(columns = ['FullBath','HalfBath', 'BsmtFullBath', 'BsmtHalfBath'])
    df['LiveableSF'] = df['1stFlrSF'] + df['2ndFlrSF']
    df['BasementSF'] = df['BsmtFinSF1'] + df['BsmtFinSF2']
    df = df.drop(columns = ['1stFlrSF','2ndFlrSF','BsmtFinSF1','BsmtFinSF2'])
    df['TotalPorchSF'] = df['OpenPorchSF'] + df['3SsnPorch'] + df['EnclosedPorch'] + df['ScreenPorch'] + df['WoodDeckSF']
    df = df.drop(columns = ['OpenPorchSF','3SsnPorch','EnclosedPorch','ScreenPorch','WoodDeckSF'])

    return df
```

Nominal Data

LotConfig: Lot configuration

Inside	Inside lot
Corner	Corner lot
CulDSac	Cul-de-sac
FR2	Frontage on 2 sides of property
FR3	Frontage on 3 sides of property

```
nominal_transformer = Pipeline(steps=[
    ('impute', SimpleImputer(strategy = 'most_frequent')),
    ('onehot_encode', OneHotEncoder(handle_unknown = 'ignore', sparse_output = False))
])
```

Ordinal Data

LotConfig: Lot configuration

Inside	Inside lot
Corner	Corner lot
CulDSac	Cul-de-sac
FR2	Frontage on 2 sides of property
FR3	Frontage on 3 sides of property

```
ordinal_transformer = Pipeline(steps=[
    ('impute', SimpleImputer(strategy = 'most_frequent')),
    ('ordinal_encode', OrdinalEncoder(handle_unknown = 'use_encoded_value', unknown_value = -1))
])
```

Numerical Data

BsmtFinSF2: Type 2 finished square feet

BsmtUnfSF: Unfinished square feet of basement area

TotalBsmtSF: Total square feet of basement area

```
numerical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy = 'median'))
])
```

```python
ordinal_transformer = Pipeline(steps=[
    ('impute', SimpleImputer(strategy = 'most_frequent')),
    ('ordinal_encode', OrdinalEncoder(handle_unknown = 'use_encoded_value', unknown_value = -1))
])

nominal_transformer = Pipeline(steps=[
    ('impute', SimpleImputer(strategy = 'most_frequent')),
    ('onehot_encode', OneHotEncoder(handle_unknown = 'ignore', sparse_output = False))
])

numerical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy = 'median'))
])

preprocessor = ColumnTransformer(
    transformers=[
        ('num', numerical_transformer, numerical_columns),
        ('ord', ordinal_transformer, ordinal_columns),
        ('nom', nominal_transformer, nominal_columns)
    ],
    remainder='passthrough'
)
```

(variable) ordinal_transformer: Pipeline

Model

```
XGB = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('regressor', XGBRegressor(random_state=13))
])

param_grid_XGB = {
    'regressor__learning_rate': [0.05, 0.1, 0.2],
    'regressor__n_estimators': [300],
    'regressor__max_depth': [3],
    'regressor__min_child_weight': [1,2,3],
    'regressor__gamma': [0, 0.1, 0.2],
    'regressor__subsample': [0.8, 0.9, 1.0],
    'regressor__colsample_bytree': [0.8, 0.9, 1.0],
}

print("Training XGBOOST model")
xgb_cv = GridSearchCV(XGB, param_grid_XGB, cv = 3, scoring='neg_mean_squared_error', n_jobs = -1)
xgb_cv.fit(X_train, y_train)
```

Training results

XGBoost performed significantly better

```
Training RandomForest model
Random forest: RMSE on training set is 0.13692830453066887
Random Forest: RMSE on test set is 0.18546344472482734
Training XGBOOST model
XGB: RMSE on training set is 0.11968649976052455
XGB: RMSE on test set is 0.16440886930260054
```

Kaggle Submission

Leaderboard: top 25% - 1527 out of 6060 submissions

House Prices - Advanced Regression Techniques

Overview Data Code Models Discussion Leaderboard Rules Team Submissions

1527	**Omar Elbasha**		0.13211

Your First Entry!
Welcome to the leaderboard!

Thank you