CatBoost Regression README (2)

The key libraries that we used in our model were:

1. **kaggle** - for authentication and to download the dataset using the Kaggle API
2. **pandas** - for loading, cleaning, and manipulating the data
3. **numpy** - for the numerical operations such as log transform
4. **scikit-learn**
    a. **train_test_spli**t - used to create training and the test sets
    b. **OneHotEncoder, ColumnTransformer, Pipeline** - imported during development stage but the main model uses **CatBoost** for handling the categorical features
    c. **RandomForestRegressor** - imported during development stage but not used in the final model
    d. **mean_absolute_error**, **r2_score** - for the model evaluation
5. **catboost**
    a. **CatBoostRegressor** - the main regression model
    b. **Pool** - CatBoost data structure that specifies categorical features

In order to allow the notebook to download the dataset directly from Kaggle, you need a Kaggle API token. You need to go to your kaggle account and then create a new API token. This will download a file named **kaggle.json** that needs to be placed in the same directory as **Regression.ipynb** when running the code. The code in the JupyterNotebook will then run:

```
!mkdir -p ~/.config/kaggle

!mv kaggle.json ~/.config/kaggle/

!chmod 600 ~/.config/kaggle/kaggle.json
```

This code then will authenticate and download the dataset:

```
import kaggle

kaggle.api.authenticate()

!kaggle datasets download -d

ahmedshahriarsakib/usa-real-estate-dataset
```

CatBoost Regression README (2)

The zip file is then extracted:

```
import zipfile

with zipfile.ZipFile("usa-real-estate-dataset.zip", "r") as zip_ref:

    zip_ref.extractall("usa-real-estate-dataset")
```

Running the Code:

Create an environment on JupyterNotebook and make sure the directory has both **Regression.ipynb** and **kaggle.json**

The install dependencies are: **%pip install kaggle pandas numpy scikit-learn catboost**

Open Regression.ipynb and run the cells in order to:

1) Install and configure packages
2) Authenticate and download the data from Kaggle
3) Extract the CSV file and load it into pandas
4) Preprocess the data and train the model
5) Evaluate the model
6) Run the interactive prediction function at the end

For feature selection and cleaning the notebook runs:

```
df_model = df[["price", "state", "city", "zip_code", "bed", "bath", "house_size"]].copy()

df_model = df_model.dropna(subset=["price", "state", "city", "zip_code", "bed", "bath", "house_size"])

df_model["zip_code"] = df_model["zip_code"].astype(str)

df_model["state"] = df_model["state"].astype(str)

df_model["city"] = df_model["city"].astype(str)
```

CatBoost Regression README (2)

Only the rows where all the selected columns are present are kept for training. The state, city, and zip-code are converted to strings to be treated as categorical features. The numerical features are bed, bath, and house_size.

In order to keep the dataset manageable, the code limits the number of examples per state to 25,000:

```
max_per_state = 25000

df_model = (
    df_model
    .groupby("state", group_keys=False)
    .apply(lambda g: g.sample(min(len(g), max_per_state), random_state=42))
    .reset_index(drop=True)
)
```

The code then modes the logarithmic value of the price to help stabilize the variance and make distributions more suitable for regression:

```
X = df_model.drop("price", axis=1)
y = np.log1p(df_model["price"])  #log(price + 1)
```

To train and test we had to cap the total dataset size to 1,100, 000 to maintain a level of efficiency:

```
N = 1_100_000
if len(X) > N:
    X_sample = X.sample(n=N, random_state=42)
    y_sample = y.loc[X_sample.index]
else:
    X_sample = X
    y_sample = y
```

CatBoost Regression README (2)

```
X_train, X_test, y_train, y_test = train_test_split(
        X_sample, y_sample, test_size=0.2, random_state=42
)
```

This code is the CatBoost Model:

```
cat_features = ["state", "city", "zip_code"]     #tells CatBoost the categorical columns

train_pool = Pool(X_train, y_train, cat_features=cat_features)
test_pool = Pool(X_test, y_test, cat_features=cat_features)

model = CatBoostRegressor(
        depth=8,
        learning_rate=0.12,
        n_estimators=500,
        loss_function="MAE",       #optimizes Mean Absolute Error
        random_seed=42,
        verbose=100                #prints the progress every 100 iterations at training
)

model.fit(train_pool, eval_set=test_pool)
```

The model is then evaluated with this code:

```
y_pred_log = model.predict(test_pool)

# Convert predictions and true values back to dollars
y_pred = np.expm1(y_pred_log)
y_true = np.expm1(y_test)
```

The mean absolute error produced was around 145,851 , meaning that on average the absolute difference between the predicted and true house price is around $146k. The R2 score was around

CatBoost Regression README (2)

0.51 which means the model was able to explain about half of the variance in house prices in the test set. For a real-world problem this is an acceptable result.

The code then creates a help function **ask_and_predict()** that prompts the user for house information and prints an estimated price:

```
def ask_and_predict():
    print("Enter house information to get an estimated price:\n")

    state = input("State: ")
    city = input("City: ")
    zip_code = input("Zip code: ")
    bed = float(input("Number of bedrooms: "))
    bath = float(input("Number of bathrooms: "))
    house_size = float(input("House size (sqft): "))

    sample = pd.DataFrame({
        "state": [state],
        "city": [city],
        "zip_code": [str(zip_code)],
        "bed": [bed],
        "bath": [bath],
        "house_size": [house_size],
        # add extra features here if df_model has them, ("acre_lot", "status")
})

sample_pool = Pool(sample, cat_features=cat_features)

# model output is log(price + 1)
predicted_log = model.predict(sample_pool)[0]

 # convert back to dollars
predicted_price = np.expm1(predicted_log)

print("\nEstimated price: ${:,.0f}".format(predicted_price))
return predicted_price
```

CatBoost Regression README (2)

Then place **ask_and_predict()** in a cell and run it.

The main files in this project are **Regression.ipynb**: used for data download, exploration, preprocessing, model training, evaluation, and interactive prediction , **kaggle.json**: must be present locally for Kaggle API access , and **usa-real-estate-dataset.zip** and its extracted folder: downloaded automatically by the code in the notebook

The external libraries and APIs used were **Kaggle API**, **CatBoost**, **scikit-learn**, **pandas**, **and numpy**.

Please refer to the original Kaggle dataset page for license details and permitted uses of the data. The dataset used was **USA Real Estate Dataset** by Ahmed Shariar Sakib on Kaggle