Pycaret Assignment

❖ Regression

❖ Kaggle Notebook :- Link

❖ Submitted By :- Dev Mulchandani

❖ Overview :-

In my regression notebook, I used PyCaret's regression module to build a model that predicts continuous numeric values. After loading the dataset and identifying the target column (in this case, SalePrice for the House Prices dataset), I initialized PyCaret using the setup() function, which handled preprocessing and feature scaling automatically. Then I compared multiple regression models with compare_models() and selected the best one based on performance metrics like R^2 and RMSE. Finally, I finalized the model, generated predictions on the test data, and exported the results to a submission.csv file. This demonstrated the complete regression workflow — from data setup to model training, evaluation, and prediction — with minimal coding using PyCaret.

❖ Screenshots :-

Regression PyCaret DEV_M Draft saved

File Edit View Run Settings Add-ons Help

Run All Markdown

PyCaret Regression — Kaggle

For Kaggle: use **Add data** to attach a dataset. CSV files appear under `/kaggle/input/...`. Run the first cell to install PyCaret 3.3.2.

Assignment Done by :- **Dev Mulchandani**

+ Code + Markdown

[1]:

```
%pip -q install -U pip
%pip -q install 'pycaret==3.3.2'
import sys
print('Python:', sys.version)
```

[2]:

```
# List CSVs under /kaggle/input and pick one by index
import glob, pandas as pd

csvs = sorted(glob.glob('/kaggle/input/**/*.csv', recursive=True))
if not csvs:
    raise SystemExit("No CSVs found. Click 'Add Data' in Kaggle and attach your dataset, then rerun.")

for i, p in enumerate(csvs):
    print(f"{i}: {p}")

idx = int(input("Enter the index of the CSV to load: "))
DATA_PATH = csvs[idx]
print("✅ Using:", DATA_PATH)

data = pd.read_csv(DATA_PATH)
print("Shape:", data.shape)
display(data.head())
```

```
0: /kaggle/input/test-regression/test.csv
1: /kaggle/input/train-regression/train.csv
Enter the index of the CSV to load:  1
✅ Using: /kaggle/input/train-regression/train.csv
Shape: (1460, 81)
```

Zoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC	Fence	MiscFeature	MiscVal	MoSold	YrSold	SaleType	SaleCondition	SalePrice
RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	0	2	2008	WD	Normal	208500
RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	0	5	2007	WD	Normal	181500
RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	9	2008	WD	Normal	223500
RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	2	2006	WD	Abnorml	140000
RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	12	2008	WD	Normal	250000

[3]:

```
# --- Run this cell FIRST (before importing pycaret.*) ---
import os
os.environ["CUDA_VISIBLE_DEVICES"] = ""   # hide GPUs so RAPIDS/GPU paths are skipped
# If RAPIDS bits are present, uninstall so PyCaret won't touch them
!pip -q uninstall -y cuml cudf cuml-cu12 cudf-cu12 cupy-cuda12x cupy-cuda11x cupy || true
```

[4]:

```
from pycaret.regression import *
TARGET = input('Enter target column name: ')
exp = setup(data, target=TARGET, session_id=123, use_gpu=False)
best = compare_models()
evaluate_model(best)
save_model(best, 'best_regression_model')
```

```
Enter target column name:  SalePrice
```

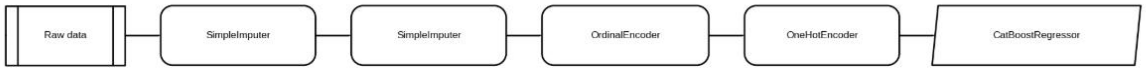
	Description	Value
0	Session id	123
1	Target	SalePrice
2	Target type	Regression
3	Original data shape	(1460, 81)
4	Transformed data shape	(1460, 279)
5	Transformed train set shape	(1021, 279)
6	Transformed test set shape	(439, 279)
7	Numeric features	37
8	Categorical features	43
9	Rows with missing values	100.0%
10	Preprocess	True
11	Imputation type	simple
12	Numeric imputation	mean
13	Categorical imputation	mode
14	Maximum one-hot encoding	25
15	Encoding method	None
16	Fold Generator	KFold
17	Fold Number	10
18	CPU Jobs	-1
19	Use GPU	False
20	Log Experiment	False

21	Experiment Name	reg-default-name
22	USI	12f3

	Model	MAE	MSE	RM
catboost	CatBoost Regressor	15387.2909	792470919.4914	271
gbr	Gradient Boosting Regressor	17276.3461	847357928.1180	283
lightgbm	Light Gradient Boosting Machine	17701.1514	1019922835.2337	310
xgboost	Extreme Gradient Boosting	19068.1370	1120869534.7547	321
rf	Random Forest Regressor	19042.9144	1117550843.2233	324
et	Extra Trees Regressor	18873.1767	1163678875.5551	327
llar	Lasso Least Angle Regression	18747.1670	1324909515.6131	337
ada	AdaBoost Regressor	25963.6459	1417400709.1888	370
ridge	Ridge Regression	20081.3998	1526032661.7945	365
en	Elastic Net	21115.3002	1782962549.6151	389
omp	Orthogonal Matching Pursuit	22617.0267	1808808421.5105	395
lasso	Lasso Regression	20570.3408	1877626177.9343	404
	Bayesian			

Plot Type:

Pipeline Plot | Hyperparameters | Residuals | Prediction Error | Cooks Distance | Feature Selection | Learning Curve | Manifold Learning | Validation Curve | Feature Importance | Feature Importanc... | Decision Tree | Interactive Residu...



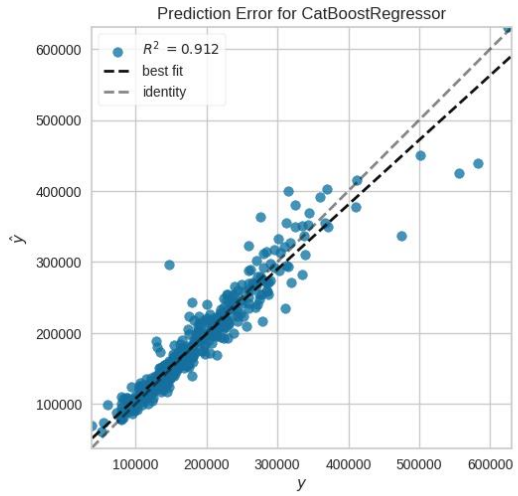

```
Transformation Pipeline and Model Successfully Saved
```

[4]:
```
(Pipeline(memory=Memory(location=None),
          steps=[('numerical_imputer',
                  TransformerWrapper(include=['Id', 'MSSubClass', 'LotFrontage',
                                              'LotArea', 'OverallQual',
                                              'OverallCond', 'YearBuilt',
                                              'YearRemodAdd', 'MasVnrArea',
                                              'BsmtFinSF1', 'BsmtFinSF2',
                                              'BsmtUnfSF', 'TotalBsmtSF',
                                              '1stFlrSF', '2ndFlrSF',
                                              'LowQualFinSF', 'GrLivArea',
                                              'BsmtFullBath', 'BsmtHalfBath',
                                              'FullBath', 'Hal...
                                                          'Exterior2nd',
                                                          'MasVnrType',
                                                          'ExterQual',
                                                          'ExterCond',
                                                          'Foundation',
                                                          'BsmtQual',
                                                          'BsmtCond',
                                                          'BsmtExposure',
                                                          'BsmtFinType1',
                                                          'BsmtFinType2',
                                                          'Heating',
                                                          'HeatingQC',
                                                          'Electrical',
                                                          'KitchenQual',
                                                          'Functional',
                                                          'FireplaceQu',
                                                          'GarageType', ...],
                                                handle_missing='return_nan',
                                                use_cat_names=True))),
```

Plot Type:

Pipeline Plot | Hyperparameters | Residuals | Prediction Error | Cooks Distance | Feature Selection | Learning Curve | Manifold Learning

Validation Curve | Feature Importance | Feature Importanc... | Decision Tree | Interactive Residu...



Transformation Pipeline and Model Successfully Saved

[4]:
```
(Pipeline(memory=Memory(location=None),
          steps=[('numerical_imputer',
                  TransformerWrapper(include=['Id', 'MSSubClass', 'LotFrontage',
                                              'LotArea', 'OverallQual',
                                              'OverallCond', 'YearBuilt',
                                              'YearRemodAdd', 'MasVnrArea',
                                              'BsmtFinSF1', 'BsmtFinSF2',
                                              'BsmtUnfSF', 'TotalBsmtSF',
```

[5]:
```
# ---- Robust prediction + submission for House Prices ----
from pycaret.regression import load_model, predict_model
import pandas as pd, glob, re

# 1) Load saved model
model = load_model("best_regression_model")

# 2) Pick your TEST csv under /kaggle/input (you already picked index 0 earlier; reuse if you like)
csvs = sorted(glob.glob('/kaggle/input/**/*.csv', recursive=True))
for i, p in enumerate(csvs):
    print(f"{i}: {p}")
try:
    idx = int(input("\nEnter the index of the TEST CSV (the file WITHOUT SalePrice): "))
except Exception:
    idx = 0
TEST_PATH = csvs[idx]
print("✅ Using:", TEST_PATH)

# 3) Read test and run predictions
test = pd.read_csv(TEST_PATH)
preds = predict_model(model, data=test)   # adds a prediction column

# 4) Figure out the prediction column name returned by PyCaret
pred_col = None
for c in preds.columns:
    if c.lower() in ("label", "prediction_label", "prediction"):
        pred_col = c
        break
if pred_col is None:
    raise KeyError(
        f"Could not find prediction column in preds. Available columns: {list(preds.columns)}"
    )

# 5) Choose / create an Id column for submission
id_col = None
# exact 'Id' first
if "Id" in test.columns:
    id_col = "Id"
else:
```

```
    # look for something that looks like an id
    candidates = [c for c in test.columns if re.search(r"\bid\b", c, flags=re.I)]
    if candidates:
        id_col = candidates[0]
        # also copy it to 'Id' for Kaggle House Prices format
        test["Id"] = test[id_col]
        id_col = "Id"
    else:
        # last resort: make a simple Id 1..N
        test["Id"] = range(1, len(test) + 1)
        id_col = "Id"

# 6) Build submission in the official format (Id + SalePrice)
submission = pd.DataFrame({
    "Id": test[id_col],
    "SalePrice": preds[pred_col]
})

submission.to_csv("submission.csv", index=False)
print("✅ submission.csv saved in /kaggle/working")
submission.head()
```

```
Transformation Pipeline and Model Successfully Loaded
0: /kaggle/input/test-regression/test.csv
1: /kaggle/input/train-regression/train.csv

Enter the index of the TEST CSV (the file WITHOUT SalePrice):  0
✅ Using: /kaggle/input/test-regression/test.csv
✅ submission.csv saved in /kaggle/working
```

[5]:

	Id	SalePrice
0	1461	119623.800204
1	1462	164549.436316
2	1463	183728.263350
3	1464	190115.144086
4	1465	187445.930368