MoP to build both a Regression model (predict monthly_bill/ARPU) and a Classification model (predict churn)

GOAL:

- Train a Linear Regression model to predict a numeric target: monthly_bill/ARPU
- Train a Logistic Regression model to predict a categorical target: churn (Yes/No)
- Compare workflows and evaluation metrics

1) Setup (once)

- Open Anaconda Navigator -> Jupyter
- Create a new folder -> ML_comparison/.
- Create a new notebook -> ML_models.ipynb (use kernel – Python Conda)

Install packages if needed:

```
pip install pandas numpy scikit-learn matplotlib seaborn
```

2) Create : Let's create a dumpy random dataset billing_churn.csv

```
import pandas as pd, numpy as np

np.random.seed(42)
n = 250
df = pd.DataFrame({
    "customer_id": range(1001, 1001+n),
    "usage_gb": np.random.gamma(5, 2, n).round(2),          # positively skewed usage
    "call_mins": np.random.normal(350, 120, n).clip(50,900).round(0),
    "complaints": np.random.poisson(0.6, n),
    "plan_type": np.random.choice(["Prepaid","Postpaid"], n, p=[0.6,0.4]),
    "region": np.random.choice(["Delhi","Mumbai","Chennai","Kolkata"], n)
})
# create a realistic bill tied to usage, calls, plan, plus noise
base = 99 + 8*df["usage_gb"] + 0.2*df["call_mins"] + 30*df["complaints"]
base += np.where(df["plan_type"]=="Postpaid", 120, 0)
df["monthly_bill"] = (base + np.random.normal(0, 40, n)).round(2)
# churn probability higher for high complaints & low usage value users
logit = -2.2 + 0.35*df["complaints"] - 0.002*(df["monthly_bill"]-200) +
0.15*(df["usage_gb"]<5)
prob = 1/(1+np.exp(-logit))
df["churn"] = np.where(np.random.rand(n)<prob, "Yes", "No")

df.to_csv("billing_churn.csv", index=False)
print("Saved billing_churn.csv with shape:", df.shape)
df.head()
```

3) Load & quick inspect

```
import pandas as pd
df = pd.read_csv("billing_churn.csv")
print(df.shape); df.head()
```

4) Basic preprocessing

- Select numeric features for a simple start.
- Encode churn to 0/1 for classification.

```
import numpy as np

num_features = ["usage_gb","call_mins","complaints"]
X_num = df[num_features].copy()

# Targets
y_reg = df["monthly_bill"]              # regression target
y_clf = (df["churn"]=="Yes").astype(int) # classification target: Yes->1, No->0
```

5) Train/test split (same split for both tasks)

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_reg_train, y_reg_test = train_test_split(
    X_num, y_reg, test_size=0.3, random_state=42)

# Use the same split indices for classification
_, _, y_clf_train, y_clf_test = train_test_split(
    X_num, y_clf, test_size=0.3, random_state=42)
```

6) Model A Linear Regression (predict monthly_bill)

Train -> Predict -> Evaluate with RMSE (lower is better).

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

reg = LinearRegression().fit(X_train, y_reg_train)
y_reg_pred = reg.predict(X_test)
rmse = np.sqrt(mean_squared_error(y_reg_test, y_reg_pred))

print("=== REGRESSION ===")
print("Features:", num_features)
```

```
print("Coefficients:", dict(zip(num_features, reg.coef_.round(3)))) # how of each of the
parameter increase will impact the billing.
print("Intercept:", round(reg.intercept_, 2))
print("RMSE:", round(rmse, 2))

import matplotlib.pyplot as plt
resid = y_reg_test - y_reg_pred
plt.scatter(y_reg_pred, resid, alpha=0.6)
plt.axhline(0, color='gray')
plt.xlabel("Predicted bill"); plt.ylabel("Residuals")
plt.title("Residuals vs Predicted (Regression)")
plt.show()

df = pd.read_csv("billing_churn.csv")

X_all = df[num_features]
df["predicted_monthly_bill"] = reg.predict(X_all).round(2)
output_path = "billing_churn_with_predictions.csv"
df.to_csv(output_path, index=False)

print(f"\n✅ Saved new file with predictions → {output_path}")
print("Preview:")
print(df.head())

7) Model B — Logistic Regression (predict churn)
Train -> Predict -> Evaluate with Accuracy, and also Precision/Recall/F1 + Confusion Matrix.

from sklearn.linear_model import LogisticRegression
from sklearn.metrics import (
    accuracy_score, precision_recall_fscore_support,
    confusion_matrix, classification_report, roc_auc_score
)
import pandas as pd
import numpy as np

# Train the classifier with class balance
clf = LogisticRegression(max_iter=200, class_weight="balanced", solver="liblinear")
clf.fit(X_train, y_clf_train)

# Predict labels & probabilities
y_clf_pred = clf.predict(X_test)
y_proba = clf.predict_proba(X_test)[:, 1]

# Evaluate standard metrics
acc = accuracy_score(y_clf_test, y_clf_pred)
prec, rec, f1, _ = precision_recall_fscore_support(
    y_clf_test, y_clf_pred, average="binary", zero_division=0
```

```
)

print("\n=== CLASSIFICATION (Balanced) ===")
print(f"Accuracy: {acc:.3f}")
print(f"Precision: {prec:.3f}  Recall: {rec:.3f}  F1: {f1:.3f}")
print("\nConfusion matrix [tn, fp; fn, tp]:\n", confusion_matrix(y_clf_test, y_clf_pred))
print("\nDetailed report:\n",
    classification_report(y_clf_test, y_clf_pred,
                          target_names=["No churn","Churn"], zero_division=0))

print(f"\n@threshold={threshold}: "
    f"Accuracy={acc_t:.3f}  Precision={prec_t:.3f}  Recall={rec_t:.3f}  F1={f1_t:.3f}")

# Reload dataset (already has predicted_monthly_bill)
df = pd.read_csv("billing_churn_with_predictions.csv")

# Predict churn probability & label for all customers
X_all = df[num_features]
df["churn_probability"] = clf.predict_proba(X_all)[:, 1].round(3)
df["predicted_churn"] = (df["churn_probability"] >= threshold).astype(int)

# Save updated dataset
output_file = "billing_churn_with_predictions_churn.csv"
df.to_csv(output_file, index=False)

print(f"\n✅ Saved file with churn predictions → {output_file}")
print(df.head())
```

8) Side-by-side takeaway

- Regression (numeric): report RMSE and interpret coefficients (how usage_gb, call_mins, complaints change the bill).
- Classification (Yes/No): report Accuracy, but discuss Recall if detecting churners is business-critical.

Simple comparison print:

```
print("\nSUMMARY")
print(f"Regression RMSE: {rmse:.2f}")
print(f"Classification Acc: {acc:.3f} | Precision: {prec:.3f} | Recall: {rec:.3f} | F1: {f1:.3f}")
```

9) What to submit (students)

- Notebook.ipynb
- billing_churn.csv and other *.csv files