# LAB ASSIGNMENT-1

2303A52183

BATCH:34

**Q1: DriveFast – Car Rental App (Simple Linear Regression + SHAP)**

```
# Q1: DriveFast – Car Rental App Analysis
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

downloads = np.array([2, 3, 1, 2, 4]).reshape(-1, 1)
bookings = np.array([40, 55, 30, 45, 65])

model = LinearRegression()
model.fit(downloads, bookings)

slope = model.coef_[0]
intercept = model.intercept_

print(f"Model: y = {intercept:.2f} + {slope:.2f}x")

baseline = np.mean(bookings)
print(f"Baseline (mean of bookings): {baseline:.2f}")

predictions = model.predict(downloads)
shap_values = predictions - baseline

results = pd.DataFrame({
    "AppDownloads (x100s)": downloads.flatten(),
    "Actual Bookings": bookings,
    "Predicted Bookings": predictions.round(2),
    "SHAP Value": shap_values.round(2),
    "Over/Under": ["Under" if p < a else "Over" if p > a else "Exact"
for p, a in zip(predictions, bookings)]
})

print(results)
```

```
Model: y = 18.85 + 11.73x
Baseline (mean of bookings): 47.00
   AppDownloads (x100s)  Actual Bookings  Predicted Bookings  SHAP
Value  \
0                     2               40               42.31       -
4.69
```

```
1                       3                 55                54.04
7.04
2                       1                 30                30.58      -
16.42
3                       2                 45                42.31       -
4.69
4                       4                 65                65.77
18.77

  Over/Under
0       Over
1      Under
2       Over
3      Under
4       Over
```

**Q2: DriveFast – Car Rental Demand (Multiple Linear Regression + SHAP)**

```python
#Q2: Multiple Linear Regression with SHAP decomposition
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression


X = pd.DataFrame({
    "FuelPrice": [90, 85, 95, 80, 92],
    "Holiday": [0, 1, 0, 1, 0]
})
y = np.array([100, 130, 90, 140, 95])

model = LinearRegression()
model.fit(X, y)

coef = model.coef_
intercept = model.intercept_
print(f"Model: y = {intercept:.2f} + {coef[0]:.2f}*FuelPrice +
{coef[1]:.2f}*Holiday")

baseline = np.mean(y)
print(f"Baseline: {baseline:.2f}")

preds = model.predict(X)
shap_values = preds - baseline
shap_fuel = coef[0] * (X["FuelPrice"] - X["FuelPrice"].mean())
shap_holiday = coef[1] * (X["Holiday"] - X["Holiday"].mean())

results = pd.DataFrame({
    "FuelPrice": X["FuelPrice"],
    "Holiday": X["Holiday"],
    "Actual Rentals": y,
```

```
    "Predicted Rentals": preds.round(2),
    "SHAP FuelPrice": shap_fuel.round(2),
    "SHAP Holiday": shap_holiday.round(2),
    "SHAP Total": shap_values.round(2),
    "Over/Under": ["Under" if p < a else "Over" if p > a else "Exact"
for p, a in zip(preds, y)]
})

print(results)
```

```
Model: y = 278.44 + -1.99*FuelPrice + 20.46*Holiday
Baseline: 111.00
   FuelPrice  Holiday  Actual Rentals  Predicted Rentals  SHAP
FuelPrice  \
0         90        0             100              99.64            -
3.18
1         85        1             130             130.03
6.75
2         95        0              90              89.70           -
13.11
3         80        1             140             139.97
16.69
4         92        0              95              95.66            -
7.15

   SHAP Holiday  SHAP Total Over/Under
0         -8.19      -11.36      Under
1         12.28       19.03       Over
2         -8.19      -21.30      Under
3         12.28       28.97      Under
4         -8.19      -15.34       Over
```

**Q3: Diabetes Dataset Regression + SHAP**

```
# Q3: Diabetes Dataset - Regression + SHAP
import numpy as np
import pandas as pd
from sklearn.datasets import load_diabetes
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

data = load_diabetes()
X = pd.DataFrame(data.data, columns=data.feature_names)
y = data.target
X_train, X_test, y_train, y_test = train_test_split(X, y,
random_state=42)

model = LinearRegression()
model.fit(X_train, y_train)
```

```
coef = model.coef_
intercept = model.intercept_

baseline = np.mean(y_train)
print(f"Baseline (mean of target): {baseline:.2f}")

y_pred = model.predict(X_test)
shap_values = y_pred - baseline
feature_shap = X_test.mul(coef, axis=1)

summary = pd.DataFrame(feature_shap.sum(axis=1), columns=["SHAP
Total"])
summary["Baseline"] = baseline
summary["Predicted"] = y_pred
summary["Actual"] = y_test
summary["Over/Under"] = ["Under" if p < a else "Over" if p > a else
"Exact" for p, a in zip(y_pred, y_test)]

print(summary.head())
```

```
Baseline (mean of target): 154.34
     SHAP Total    Baseline   Predicted  Actual Over/Under
287  -13.716087  154.344411  137.949089   219.0      Under
211   30.868178  154.344411  182.533354    70.0       Over
72   -21.812222  154.344411  129.852954   202.0      Under
321  140.897917  154.344411  292.563092   230.0       Over
73   -26.797293  154.344411  124.867882   111.0       Over
```

**Q4: Student Performance Dataset Regression + SHAP**

```
import pandas as pd
import zipfile
import urllib.request
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import numpy as np

url =
"https://archive.ics.uci.edu/ml/machine-learning-databases/00320/stude
nt.zip"
urllib.request.urlretrieve(url, "student.zip")

with zipfile.ZipFile("student.zip", 'r') as zip_ref:
    zip_ref.extractall()

df = pd.read_csv("student-mat.csv", sep=';')

features = ['studytime', 'failures', 'absences', 'Medu', 'Fedu']
target = 'G3'
```

```
X = df[features]
y = df[target]

X_train, X_test, y_train, y_test = train_test_split(X, y,
random_state=42)

model = LinearRegression()
model.fit(X_train, y_train)

coefs = model.coef_
intercept = model.intercept_
baseline = y_train.mean()

print(f"\nModel: G3 = {intercept:.2f} + " + " + ".join([f"{c:.2f}*{f}"
for c, f in zip(coefs, features)]))
print(f"Baseline (mean final grade G3): {baseline:.2f}")

y_pred = model.predict(X_test)

shap_contributions = X_test * coefs
shap_total = shap_contributions.sum(axis=1)

results = pd.DataFrame({
    "Actual G3": y_test.values,
    "Predicted G3": y_pred,
    "SHAP Total": shap_total,
    "Baseline": baseline,
    "Over/Under": ["Under" if pred < actual else "Over" if pred >
actual else "Exact"
                   for pred, actual in zip(y_pred, y_test)]
})

for i, feature in enumerate(features):
    results[f"SHAP_{feature}"] = shap_contributions[feature]

pd.set_option("display.max_columns", None)
print(results.head(10))


Model: G3 = 9.43 + 0.36*studytime + -2.07*failures + 0.03*absences +
0.65*Medu + -0.41*Fedu
Baseline (mean final grade G3): 10.39
     Actual G3  Predicted G3  SHAP Total   Baseline Over/Under  \
78          10      4.525151   -4.902063  10.385135      Under
371         12      9.702657    0.275443  10.385135      Under
248          5      9.030174   -0.397040  10.385135       Over
55          10     11.271734    1.844520  10.385135       Over
390          9      6.809595   -2.617619  10.385135      Under
223         13     10.622006    1.194792  10.385135      Under
42          18     11.161360    1.734146  10.385135      Under
```

```
234          6     10.918486    1.491272  10.385135       Over
316          0     11.033352    1.606139  10.385135       Over
116         14     11.161360    1.734146  10.385135      Under

     SHAP_studytime  SHAP_failures  SHAP_absences  SHAP_Medu
SHAP_Fedu
78         0.357517      -6.210280       0.059595   1.302451  -
0.411346
371        0.357517      -0.000000       0.089393   0.651225  -
0.822692
248        0.715034      -2.070093       0.238382   1.953676  -
1.234038
55         0.715034      -0.000000       0.238382   1.302451  -
0.411346
390        0.715034      -4.140187       0.327775   1.302451  -
0.822692
223        0.715034      -0.000000       0.000000   1.302451  -
0.822692
42         0.715034      -0.000000       0.059595   2.604902  -
1.645385
234        0.715034      -0.000000       0.536359   0.651225  -
0.411346
316        0.715034      -0.000000       0.000000   1.302451  -
0.411346
116        0.715034      -0.000000       0.059595   2.604902  -
1.645385
```