Lab 5 solution key

1.

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from sklearn.preprocessing import PolynomialFeatures

# (a) Load dataset
data = {
    "Engine Size": [1.0, 1.3, 1.5, 1.8, 2.0, 2.3, 2.5, 3.0, 3.5, 4.0],
    "MPG": [42, 39, 35, 31, 29, 26, 24, 20, 18, 15],
}
df = pd.DataFrame(data)
X = df[["Engine Size"]]
y = df["MPG"]

# (b) Fit Linear Regression
lin_reg = LinearRegression()
lin_reg.fit(X, y)
y_lin_pred = lin_reg.predict(X)

# (c) Fit Polynomial Regression (degree 3)
poly = PolynomialFeatures(degree=3)
X_poly = poly.fit_transform(X)
poly_reg = LinearRegression()
poly_reg.fit(X_poly, y)
y_poly_pred = poly_reg.predict(X_poly)

# (d) Predict for 2.2
X_test = np.array([[2.2]])
lin_22 = lin_reg.predict(X_test)[0]
poly_22 = poly_reg.predict(poly.transform(X_test))[0]
print(f"Prediction for 2.2 -> Linear: {lin_22:.2f}, Poly: {poly_22:.2f}")

# (e) Evaluate R2
print(f"Linear R2: {r2_score(y, y_lin_pred):.4f}")
print(f"Poly R2:   {r2_score(y, y_poly_pred):.4f}")

# Visualization
plt.scatter(X, y, color="blue", label="Data")
plt.plot(X, y_lin_pred, color="red", label="Linear")
```

```
plt.plot(X, y_poly_pred, color="green", label="Poly Deg 3")
plt.xlabel("Engine Size")
plt.ylabel("MPG")
plt.legend()
plt.show()
```

2.

```
import matplotlib.pyplot as plt
import numpy as np
from scipy.stats import beta

theta = np.linspace(0, 1, 100)
# Prior Beta(1,1)
prior = beta.pdf(theta, 1, 1)

# Likelihood (10 tosses, 7 heads -> h=7, t=3)
h, t = 7, 3
likelihood = theta**h * (1 - theta) ** t

# Posterior Beta(1+7, 1+3) = Beta(8, 4)
posterior = beta.pdf(theta, 1 + h, 1 + t)

plt.plot(theta, prior, label="Prior Beta(1,1)")
plt.plot(
    theta,
    likelihood / np.trapz(likelihood, theta),
    label="Normalized Likelihood",
)
plt.plot(theta, posterior, label="Posterior Beta(8,4)")
plt.xlabel("theta")
plt.legend()
plt.show()
```

3.

```
from sklearn.datasets import load_iris
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB

# Load data
iris = load_iris()
X_tr, X_te, y_tr, y_te = train_test_split(
    iris.data, iris.target, test_size=0.2, random_state=42
)
```

```
# Train GaussianNB
clf = GaussianNB()
clf.fit(X_tr, y_tr)

# Predict & Evaluate
preds = clf.predict(X_te)
print("Accuracy:", accuracy_score(y_te, preds))
```

4.

```
# Conditional probability tables
# P(R)
p_R = 0.2
p_nR = 1 - p_R

# P(S|R)
p_S_given_R = 0.01
p_S_given_nR = 0.4

# P(W|S,R)
p_W = {
    (1, 1): 0.99,  # S=1, R=1
    (1, 0): 0.90,  # S=1, R=0
    (0, 1): 0.80,  # S=0, R=1
    (0, 0): 0.00,  # S=0, R=0
}

# (a) Compute joint probability P(R, S, W) for all states
joint = {}
for r in [0, 1]:
    pr = p_R if r else p_nR
    for s in [0, 1]:
        ps = (p_S_given_R if r else p_S_given_nR) if s else (1 - (p_S_given_R if r else
p_S_given_nR))
        for w in [0, 1]:
            pw = p_W[(s, r)] if w else (1 - p_W[(s, r)])
            joint[(r, s, w)] = pr * ps * pw

print("Joint Probabilities P(R, S, W):")
for k, v in joint.items():
    print(f"P(R={k[0]}, S={k[1]}, W={k[2]}) = {v:.5f}")

# (b) Marginal probability P(W = 1)
p_W1 = sum(v for k, v in joint.items() if k[2] == 1)
```

```
print(f"\nMarginal P(W=1): {p_W1:.5f}")

# (c) Conditional probability P(R=1 | W=1)
p_R1_W1 = sum(v for k, v in joint.items() if k[0] == 1 and k[2] == 1) / p_W1
print(f"Conditional P(R=1 | W=1): {p_R1_W1:.5f}")
```

5.

```
p_D = 0.01
p_nD = 0.99

p_pos_given_D = 0.95
p_pos_given_nD = 0.10

# P(T=+)
p_pos = (p_pos_given_D * p_D) + (p_pos_given_nD * p_nD)

# P(D | T=+)
p_D_given_pos = (p_pos_given_D * p_D) / p_pos

print(f"P(D) = {p_D}")
print(f"P(D | T=+) = {p_D_given_pos:.4f}")
```