MATH 4/5388: Machine Learning Methods

Homework 2

Due date: Tuesday, February 18

Submission Instruction:

- Submit both the Jupyter notebook file (.ipynb) and a PDF copy of the notebook.
- Ensure that your notebook runs properly before submitting it:
  - Kernel -> Restart & Run All to ensure that there are no errors.
- To generate a PDF of your notebook:
  - File -> Print Preview followed by printing to a PDF from your browser;

  or:
  - File -> Download as -> PDF via LaTeX.
- If this doesn't work, try first exporting as an HTML file and then converting that to PDF (load it in a web browser and print it to PDF).

Problem 1 (20 points)

The goal of this problem is to ensure that you understand the mathematical principles behind linear regression. Assume that we are given a training dataset consisting of inputs $\mathbf{x}_n$ and targets y_n, $n = 1, \ldots, N$. Each $\mathbf{x}_n$ is a D-dimensional vector consisting of elements $(x_{n,1}, x_{n,2}, \ldots, x_{n,D})$ and $y_n \in \mathbb{R}$.

1a. Linear Regression Prediction Function

- Explain the form of the prediction function for a linear regression model $f(\mathbf{x}_n; \boldsymbol{\theta})$. Assume we have a set of coefficients represented by a D-dimensional vector $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_D)$. For simplicity, we ignore the bias or intercept term of the linear regression model in this analysis.

1b. Residual Sum of Squares (RSS) in Compact Form

- Substitute the prediction function from **1a** into the Residual Sum of Squares (RSS) derived in Homework 1. Show that this expression can be rewritten in the form of matrices and vectors, eliminating the explicit summation.

$$\mathrm{RSS}(\boldsymbol{\theta}) = \sum_{n=1}^{N} \left(y_n - f(\mathbf{x}_n; \boldsymbol{\theta})\right)^2$$

Problem 1:

a) $f(x_n; \theta) = \sum_{d=1}^{D} x_{nd}\,\theta_d$

We sum each weight θ by the corresponding element in x_n, for each $d \in D$. Equivalentally, $\sum_{d=1}^{D} x_{nd}\theta_d =$

$$[x_{n1}, x_{n2}, \ldots, x_{nD}] \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_d \end{bmatrix} = x_n\theta,$$ representing our model's prediction for each $n \in N$.

b) $$RSS(\theta) = \sum_{n=1}^{N} (y_n - \sum_{d=1}^{D} x_{nd}\theta_d)^2$$

$$= \sum_{n=1}^{N} (y_n - x_n\theta)^2$$

$$= (y - x\theta)^T (y - x\theta)$$

Problem 2 (20 points)

In this problem, you will work with the **mtcars** dataset, which is loaded for you in the provided cell.

2a. Data Splitting

- Split the dataset into training and testing sets.
- Use 'hp' (horsepower) and 'wt' (weight) as input features, and 'mpg' (miles per gallon) as the target variable.
- After splitting, print the size of the training and testing sets.

2b. Model Fitting

Using scikit-learn, fit a linear regression model to the training data for the following four cases:

- Case 1: Use 'hp' (horsepower) as the only input feature.

- Case 2: Use 'wt' (weight) as the only input feature.
- Case 3: Use both 'hp' and 'wt' as input features without any preprocessing.
- Case 4: Use both 'hp' and 'wt' as input features with preprocessing. Choose one preprocessing technique from `sklearn.preprocessing` (e.g., `StandardScaler`, `MinMaxScaler`, `PolynomialFeatures`, etc.) and apply it before fitting the model.

2c. Model Evaluation

- Evaluate the performance of each model on the test set using at least two different metrics from `sklearn.metrics`.
- Present the evaluation results clearly for each case.

2d. Analysis

- Compare the performance of Case 3 (without preprocessing) and Case 4 (with preprocessing).
- Discuss whether preprocessing improved the model's performance when using both features. Support your answer using the evaluation metrics.

In [5]:
```
import pandas as pd

# load mtcars
df = pd.read_csv('mtcars.csv')
df.head()
```

Out[5]:

	model	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
0	Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
1	Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
2	Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
3	Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
4	Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2

In [6]:
```
# 2a
from sklearn.model_selection import train_test_split
```

```
input_features = ['hp', 'wt']
target = 'mpg'

X_train, X_test, y_train, y_test = train_test_split(df[input_features], df[target],
test_size=0.25,
random_state=5)

print(f'X_train size: {X_train.shape[0]}')
print(f'y_train size: {y_train.shape[0]}')
print(f'X_test size: {X_test.shape[0]}')
print(f'y_test size: {y_test.shape[0]}')
```

```
X_train size: 24
y_train size: 24
X_test size: 8
y_test size: 8
```

In [7]:

```
# 2b
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from sklearn.preprocessing import StandardScaler, MinMaxScaler

# Case 1
X_train, X_test, y_train, y_test = train_test_split(df[[input_features[0]]], df[target],
test_size=0.25,
random_state=5)

model1 = LinearRegression()
model1.fit(X_train, y_train)
y_pred1 = model1.predict(X_test)

# Case 2
X_train, X_test, y_train, y_test = train_test_split(df[[input_features[1]]], df[target],
test_size=0.25,
random_state=5)

model2 = LinearRegression()
model2.fit(X_train, y_train)
y_pred2 = model2.predict(X_test)

# Case 3
X_train, X_test, y_train, y_test = train_test_split(df[input_features], df[target],
```

```
test_size=0.25,
random_state=5)

model3 = LinearRegression()
model3.fit(X_train, y_train)
y_pred3 = model3.predict(X_test)

# Case 4
X_train, X_test, y_train, y_test = train_test_split(df[input_features], df[target],
test_size=0.25,
random_state=5)

scaler = StandardScaler()
standardized = scaler.fit_transform(X_train)
X_train_scale = scaler.transform(X_train)
X_test_scale = scaler.transform(X_test)

model4 = LinearRegression()
model4.fit(X_train_scale, y_train)
y_pred4 = model4.predict(X_test_scale)

# Case 4 2.0
X_train, X_test, y_train, y_test = train_test_split(df[input_features], df[target],
test_size=0.25,
random_state=5)

scaler = MinMaxScaler()
scaler.fit(X_train)
X_train_scale = scaler.fit_transform(X_train)
X_test_scale = scaler.transform(X_test)

model5 = LinearRegression()
model5.fit(X_train_scale, y_train)
y_pred5 = model5.predict(X_test_scale)
```

In [8]:

```
# 2c
from sklearn.metrics import r2_score, mean_squared_error

y_pred = [y_pred1, y_pred2, y_pred3, y_pred4, y_pred5]

for i in range(5):
  print(f'Metrics for case {i + 1}')
```

```
  print(f'R2 score: {r2_score(y_test, y_pred[i])}')
  print(f'Mean squared error: {mean_squared_error(y_test, y_pred[i])}')
```

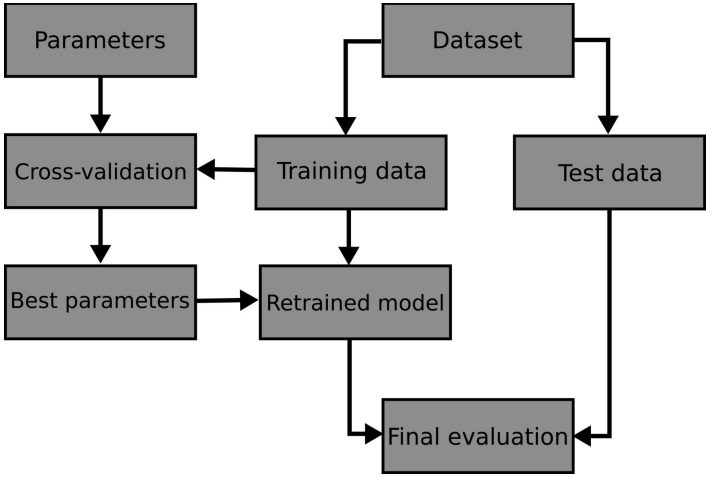
```
Metrics for case 1
R2 score: 0.5299294117029201
Mean squared error: 13.079053082601451
Metrics for case 2
R2 score: 0.687880928508651
Mean squared error: 8.684274246802499
Metrics for case 3
R2 score: 0.701160120459662
Mean squared error: 8.314799404629298
Metrics for case 4
R2 score: 0.701160120459662
Mean squared error: 8.3147994046293
Metrics for case 5
R2 score: 0.7011601204596621
Mean squared error: 8.314799404629296
```

2d.

Preprocessing didn't improve the performance of the model as seen from the metrics for cases 3-5 which are all identical. Using both features had an impact, as the R2 score for individual features was 0.53 and 0.69, while using both features was 0.70.

Problem 3 (30 points)

In this problem, you will use the California Housing dataset from `sklearn.datasets` to practice model selection and cross-validation techniques. Your goal is to predict housing prices based on various features in the dataset. The following figure illustrates the general approach that we are taking in this problem.



3a. Load and Explore the Data

- Explore the dataset: display the first few rows, summarize the features and target variable (e.g., number of samples N, number of features D, etc.)
- Split the data into training and testing sets.

3b. Define a Pipeline

- Create a pipeline that includes preprocessing step (standardize the input features using StandardScaler), use PolynomialFeatures to transform the features, and followed by `SGDRegressor` (https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDRegressor.html).

3c. Cross-Validation for Model Selection

- Use cross-validation on the training set to determine the best degree for the polynomial features.
- Test at least degrees 1 through 4.
- Evaluate model performance using Mean Squared Error (MSE) as the scoring metric.
- Report the average cross-validation score for each polynomial degree.

3d. Final Model Evaluation

- Based on your cross-validation results, select the polynomial degree with the best performance.
- Retrain the pipeline using the entire training set with the selected degree.

- Evaluate the final model on the test set using the following metrics from `sklearn.metrics`: Mean Squared Error (MSE) and R-squared.
- How well does the model generalize to unseen data?

In [11]:
```
from sklearn.datasets import fetch_california_housing

housing = fetch_california_housing()

housing.keys()
```

Out[11]:
```
dict_keys(['data', 'target', 'frame', 'target_names', 'feature_names', 'DESCR'])
```

In [12]:
```
# 3a
features = housing.data
target = housing.target
print(housing.DESCR)
print(pd.DataFrame(housing.data, columns=housing.feature_names).describe)


X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.25, random_state=4)
```

```
.. _california_housing_dataset:

California Housing dataset
--------------------------

**Data Set Characteristics:**

:Number of Instances: 20640

:Number of Attributes: 8 numeric, predictive attributes and the target

:Attribute Information:
    - MedInc        median income in block group
    - HouseAge      median house age in block group
    - AveRooms      average number of rooms per household
    - AveBedrms     average number of bedrooms per household
    - Population    block group population
    - AveOccup      average number of household members
    - Latitude      block group latitude
    - Longitude     block group longitude

:Missing Attribute Values: None

This dataset was obtained from the StatLib repository.
https://www.dcc.fc.up.pt/~ltorgo/Regression/cal_housing.html

The target variable is the median house value for California districts,
expressed in hundreds of thousands of dollars ($100,000).

This dataset was derived from the 1990 U.S. census, using one row per census
block group. A block group is the smallest geographical unit for which the U.S.
Census Bureau publishes sample data (a block group typically has a population
of 600 to 3,000 people).

A household is a group of people residing within a home. Since the average
number of rooms and bedrooms in this dataset are provided per household, these
columns may take surprisingly large values for block groups with few households
and many empty houses, such as vacation resorts.

It can be downloaded/loaded using the
:func:`sklearn.datasets.fetch_california_housing` function.
```

```
.. topic:: References

    - Pace, R. Kelley and Ronald Barry, Sparse Spatial Autoregressions,
      Statistics and Probability Letters, 33 (1997) 291-297

<bound method NDFrame.describe of        MedInc  HouseAge  AveRooms  AveBedrms  Population  AveOccup  Latitude  \
0      8.3252      41.0  6.984127   1.023810       322.0  2.555556     37.88
1      8.3014      21.0  6.238137   0.971880      2401.0  2.109842     37.86
2      7.2574      52.0  8.288136   1.073446       496.0  2.802260     37.85
3      5.6431      52.0  5.817352   1.073059       558.0  2.547945     37.85
4      3.8462      52.0  6.281853   1.081081       565.0  2.181467     37.85
...       ...       ...       ...        ...         ...       ...       ...
20635  1.5603      25.0  5.045455   1.133333       845.0  2.560606     39.48
20636  2.5568      18.0  6.114035   1.315789       356.0  3.122807     39.49
20637  1.7000      17.0  5.205543   1.120092      1007.0  2.325635     39.43
20638  1.8672      18.0  5.329513   1.171920       741.0  2.123209     39.43
20639  2.3886      16.0  5.254717   1.162264      1387.0  2.616981     39.37

       Longitude
0         -122.23
1         -122.22
2         -122.24
3         -122.25
4         -122.25
...           ...
20635     -121.09
20636     -121.21
20637     -121.22
20638     -121.32
20639     -121.24

[20640 rows x 8 columns]>
```

In [13]:

```python
# 3b/c
import numpy as np
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import SGDRegressor
from sklearn.model_selection import cross_val_score

best_degree = 1
best_score = -np.inf
```

```
for degrees in range(1, 6):
  pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('poly', PolynomialFeatures(degree = degrees)),
    ('sgd_reg', SGDRegressor(alpha=0.01, tol=0.0001, eta0=0.001, max_iter=10000))
])

  pipeline.fit(X_train, y_train)

  cv_scores = cross_val_score(pipeline, X_train, y_train, cv=5, scoring='neg_mean_squared_error')
  avg_cv = np.mean(cv_scores)


  if avg_cv > best_score:
    best_score = avg_cv
    best_degree = degrees

  print(f'Average score for degree {degrees}: {avg_cv}')

print(f'Best degree: {best_degree}')
```

```
Average score for degree 1: -0.5320564118463933
Average score for degree 2: -7.497742208565386e+26
Average score for degree 3: -5.046291521646477e+35
Average score for degree 4: -8.808015554904268e+43
Average score for degree 5: -6.854005066072789e+51
Best degree: 1
```

In [14]:
```
# 3d
pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('poly', PolynomialFeatures(degree = 1)),
    ('sgd_reg', SGDRegressor())
])

pipeline.fit(X_train, y_train)
y_pred = pipeline.predict(X_test)


print(f'R2 score: {r2_score(y_test, y_pred)}')
print(f'Mean squared error: {mean_squared_error(y_test, y_pred)}')
```

```
R2 score: 0.5077250925680745
Mean squared error: 0.6444655553035359
```

The scores reflect the model did moderately well generalizing unseen data, especially considering the size of the data/test sets.

Problem 4 (30 points)

In this problem, you will explore the relationship between explained variance and the R^2 score. You will start with empirical experiments [4388 and 5388] to observe different scenarios and then proceed to a theoretical proof [5388 only]. Recall the definitions for explained variance and the R^2 score given a set of ground-truth outputs $y_1, \ldots, y_N$ and predicted values $\hat{y}_1, \ldots, \hat{y}_N$:

$$\text{EV} = 1 - \frac{\text{Var}(y - \hat{y})}{\text{Var}(y)}$$

$$R^2 = 1 - \frac{\sum_{n=1}^{N}(y_n - \hat{y}_n)^2}{\sum_{i=n}^{N}(y_n - \bar{y})^2}$$

4a. Empirical analysis [4388 and 5388]

- Write a Python script to explore the relationship between explained variance and the R^2 score.
- Provide different scenarios in your experiment:
  - Scenario 1: Perfect predictions.
  - Scenario 2: Predictions with zero-mean residuals.
  - Scenario 3: Predictions with biased residuals.
- For each scenario, use `sklearn.metrics.explained_variance_score` and `sklearn.metrics.r2_score`.

4b. Theoretical analysis [5388]

- Prove under what conditions the explained variance reduces to the R^2 score.

In [17]:
```
# 4a
from sklearn.metrics import explained_variance_score
```

```
# Scenario 1:
y_true = y_pred = np.arange(0, 11)
print(f'Explained variance for case 1: {explained_variance_score(y_true, y_pred)}')
print(f'R2 score for case 1: {r2_score(y_true, y_pred)}')

# Scenario 2:
y_true = np.arange(0, 11)
y_pred = [5] * 11
print(f'Explained variance for case 2: {explained_variance_score(y_true, y_pred)}')
print(f'R2 score for case 2: {r2_score(y_true, y_pred)}')

# Scenario 3:
y_true = np.arange(0, 11)
y_pred = [50, -50, 40, -40, 30, -30, 20, -20, 10, -10, 0]
print(f'Explained variance for case 3(alternating): {explained_variance_score(y_true, y_pred)}')
print(f'R2 score for case 3: {r2_score(y_true, y_pred)}')

y_true = np.arange(0, 11)
y_pred = [-10] * 11
print(f'Explained variance for case 3(-10s): {explained_variance_score(y_true, y_pred)}')
print(f'R2 score for case 3: {r2_score(y_true, y_pred)}')

y_true = np.arange(0, 11)
y_pred = [200] * 11
print(f'Explained variance for case 3(200s): {explained_variance_score(y_true, y_pred)}')
print(f'R2 score for case 3: {r2_score(y_true, y_pred)}')
```

```
Explained variance for case 1: 1.0
R2 score for case 1: 1.0
Explained variance for case 2: 0.0
R2 score for case 2: 0.0
Explained variance for case 3(alternating): -102.72727272727272
R2 score for case 3: -105.22727272727273
Explained variance for case 3(-10s): 0.0
R2 score for case 3: -22.5
Explained variance for case 3(200s): 0.0
R2 score for case 3: -3802.5
```