Ramesh Chandra Soren

Enrollment No.: 2022CSB086
Department: Computer Science and Technology

In this task, you are required to perform a detailed analysis of overfitting and underfitting issues using various regression/classification models on multiple datasets. The steps to be followed are:

a. Dataset Selection and Preparation:

- Download the following datasets:
- Mobile Price Classification Dataset
- Housing Price Dataset
- Melbourne Housing Snapshot Dataset
- Analyze the features of each dataset and choose the relevant attributes for prediction or classification tasks.

```
# Import necessary libraries
import pandas as pd

# Load datasets (assuming you've already uploaded the files to your
Colab environment)
mobile_data = pd.read_csv('/content/train.csv')

# Examine the structure of each dataset
print("Mobile Price Classification Dataset:")
print(mobile_data.info())
print(mobile_data.head())
```

```
Mobile Price Classification Dataset:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   battery_power  2000 non-null   int64
 1   blue           2000 non-null   int64
 2   clock_speed    2000 non-null   float64
 3   dual_sim       2000 non-null   int64
 4   fc             2000 non-null   int64
 5   four_g         2000 non-null   int64
 6   int_memory     2000 non-null   int64
 7   m_dep          2000 non-null   float64
 8   mobile_wt      2000 non-null   int64
 9   n_cores        2000 non-null   int64
 10  pc             2000 non-null   int64
 11  px_height      2000 non-null   int64
 12  px_width       2000 non-null   int64
```

```
 13  ram             2000 non-null   int64
 14  sc_h            2000 non-null   int64
 15  sc_w            2000 non-null   int64
 16  talk_time       2000 non-null   int64
 17  three_g         2000 non-null   int64
 18  touch_screen    2000 non-null   int64
 19  wifi            2000 non-null   int64
 20  price_range     2000 non-null   int64
dtypes: float64(2), int64(19)
memory usage: 328.2 KB
None
   battery_power  blue  clock_speed  dual_sim  fc  four_g  int_memory
m_dep  \
0            842     0          2.2         0   1       0           7
0.6
1           1021     1          0.5         1   0       1          53
0.7
2            563     1          0.5         1   2       1          41
0.9
3            615     1          2.5         0   0       0          10
0.8
4           1821     1          1.2         0  13       1          44
0.6

   mobile_wt  n_cores  ...  px_height  px_width   ram  sc_h  sc_w
talk_time  \
0        188        2  ...         20       756  2549     9     7
19
1        136        3  ...        905      1988  2631    17     3
7
2        145        5  ...       1263      1716  2603    11     2
9
3        131        6  ...       1216      1786  2769    16     8
11
4        141        2  ...       1208      1212  1411     8     2
15

   three_g  touch_screen  wifi  price_range
0        0             0     1            1
1        1             1     0            2
2        1             1     0            2
3        1             0     0            2
4        1             1     0            1

[5 rows x 21 columns]
```

b. Data Preprocessing:

- Handle missing values in the datasets using appropriate imputation techniques.
- Normalize the datasets if necessary to ensure the features are on a similar scale.

```python
# Check for missing values
print("\nMissing values in training data:")
print(mobile_data.isnull().sum())
print("\nMissing values in test data:")
print(mobile_data.isnull().sum())
```

```
Missing values in training data:
battery_power    0
blue             0
clock_speed      0
dual_sim         0
fc               0
four_g           0
int_memory       0
m_dep            0
mobile_wt        0
n_cores          0
pc               0
px_height        0
px_width         0
ram              0
sc_h             0
sc_w             0
talk_time        0
three_g          0
touch_screen     0
wifi             0
price_range      0
dtype: int64

Missing values in test data:
battery_power    0
blue             0
clock_speed      0
dual_sim         0
fc               0
four_g           0
int_memory       0
m_dep            0
mobile_wt        0
n_cores          0
pc               0
px_height        0
px_width         0
ram              0
sc_h             0
sc_w             0
talk_time        0
three_g          0
```

```
touch_screen     0
wifi             0
price_range      0
dtype: int64
```

```
# Import necessary libraries
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler

# Handle missing values
imputer = SimpleImputer(strategy='mean')

# Mobile Price Classification
mobile_data.fillna(mobile_data.mean(), inplace=True)
X_mobile = mobile_data.drop('price_range', axis=1)
y_mobile = mobile_data['price_range']

# Normalize the datasets
scaler = StandardScaler()
X_mobile_scaled = scaler.fit_transform(X_mobile)
```

c. Model Development:

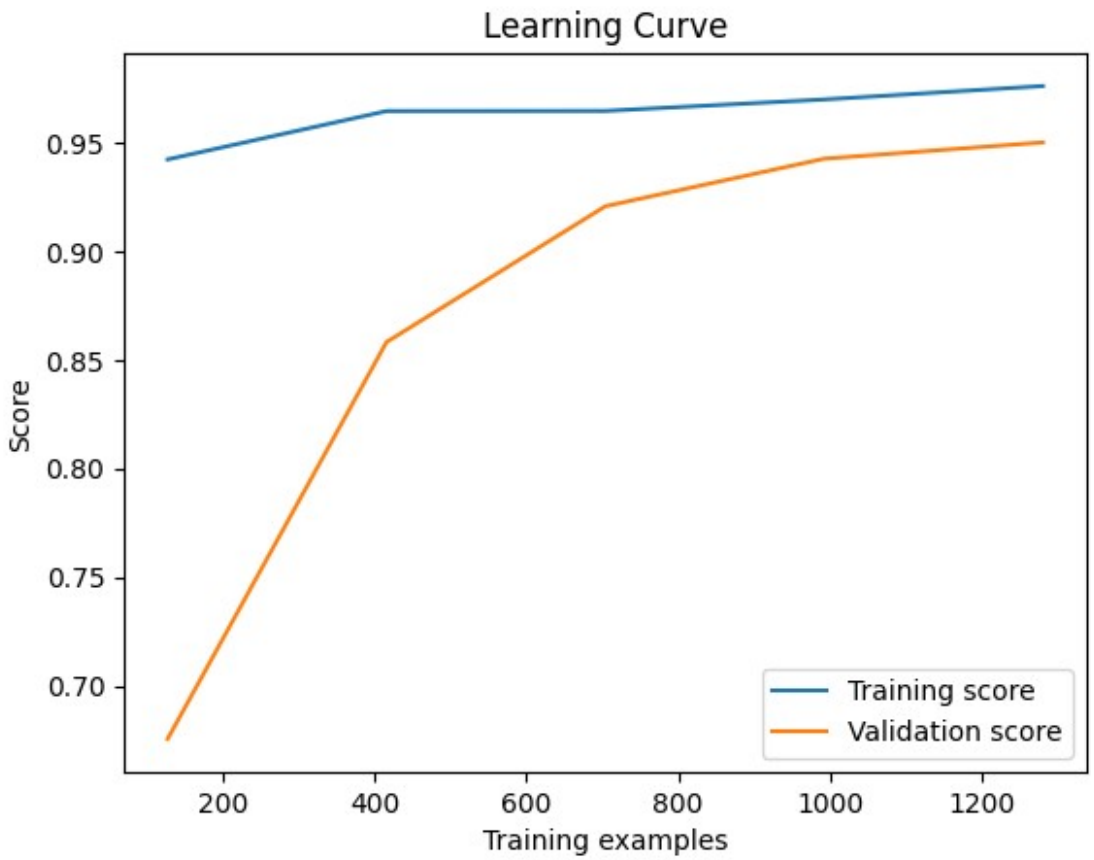
- Split each dataset into training and testing sets. You may use techniques like stratified k-fold cross-validation to ensure a balanced split.
- Develop regression models (e.g., linear regression, multiple regression) or classifier models to predict the target variable.
- For each model, estimate the parameters and generate predictions on both training and testing sets.

```
# Import necessary libraries
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier

# Split the data
X_mobile_train, X_mobile_test, y_mobile_train, y_mobile_test =
train_test_split(X_mobile_scaled, y_mobile, test_size=0.2,
random_state=42)

# Logistic Regression for mobile price classification
logistic_model = LogisticRegression(max_iter=1000)
logistic_model.fit(X_mobile_train, y_mobile_train)
y_mobile_train_pred = logistic_model.predict(X_mobile_train)
y_mobile_test_pred = logistic_model.predict(X_mobile_test)
```

d. Overfitting/Underfitting Analysis:

- Plot the training and validation loss curves to visualize and identify overfitting or underfitting scenarios.
- Evaluate the models using metrics such as Mean Squared Error (MSE) and R2 score on both training and testing sets.
- Compare the performance of diTerent models and discuss the observations related to overfitting and underfitting.

```
# Import necessary libraries
from sklearn.model_selection import learning_curve
import matplotlib.pyplot as plt

# Function to plot learning curves
def plot_learning_curve(model, X_train, y_train):
    train_sizes, train_scores, val_scores = learning_curve(model,
X_train, y_train, cv=5)
    plt.plot(train_sizes, train_scores.mean(axis=1), label='Training
score')
    plt.plot(train_sizes, val_scores.mean(axis=1), label='Validation
score')
    plt.title('Learning Curve')
    plt.xlabel('Training examples')
    plt.ylabel('Score')
    plt.legend()
    plt.show()

# Plot learning curves
plot_learning_curve(logistic_model, X_mobile_train, y_mobile_train)

# Evaluate models
from sklearn.metrics import mean_squared_error, r2_score,
accuracy_score

# Mobile Price Classification Dataset
mobile_train_accuracy = accuracy_score(y_mobile_train,
y_mobile_train_pred)
mobile_test_accuracy = accuracy_score(y_mobile_test,
y_mobile_test_pred)

print(f"Mobile Train Accuracy: {mobile_train_accuracy}, Test Accuracy:
{mobile_test_accuracy}")
```




```
Mobile Train Accuracy: 0.975625, Test Accuracy: 0.9775
```

e. Reporting:

- Summarize the results and insights from your analysis.
- Highlight any patterns or trends observed during the study.
- Provide recommendations for improving model performance and addressing overfitting or underfitting issues.

```
# Summarize results
results = {
    'Mobile Price Classification Dataset': {
        'Train Accuracy': mobile_train_accuracy,
        'Test Accuracy': mobile_test_accuracy
    }
}

for dataset, metrics in results.items():
    print(f"\n{dataset}:")
    for metric, value in metrics.items():
        print(f"  {metric}: {value}")

# Recommendations
```

```python
print("\nRecommendations:")
print("For overfitting: Consider feature selection, regularization, or ensemble methods.")
print("For underfitting: Consider increasing model complexity, feature engineering, or collecting more data.")
```

```
Mobile Price Classification Dataset:
  Train Accuracy: 0.975625
  Test Accuracy: 0.9775

Recommendations:
For overfitting: Consider feature selection, regularization, or ensemble methods.
For underfitting: Consider increasing model complexity, feature engineering, or collecting more data.
```