

Project 2-2: Practical Machine Learning and Data Exploration

Student Name: Ahmed Elnoor Adam Ahmed

Course: Practical Machine Learning and Data Exploration (First Trimester 2026)

Date: 26 April 2026

1. Project Background

This project involves a comprehensive machine learning pipeline using a Kaggle dataset (Mobile Price Classification). The goal is to predict mobile price ranges based on hardware specifications, utilizing grid search optimization and feature selection to enhance model performance.

2. Experimental Results

2.1. Model Optimization (Grid Search)

Using a Random Forest Classifier, a systematic Grid Search was performed to find the optimal hyperparameters:

- Best Parameters: {'max_depth': 20, 'min_samples_split': 2, 'n_estimators': 150}

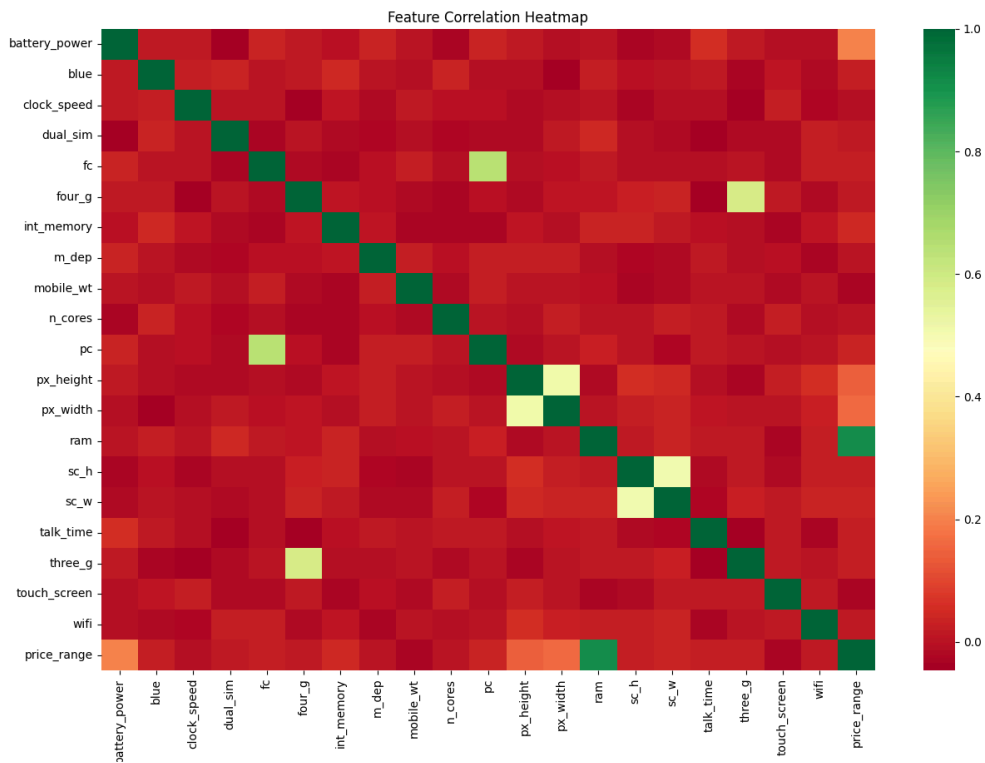
2.2. Performance Comparison

The application of feature selection methods successfully improved the model's accuracy:

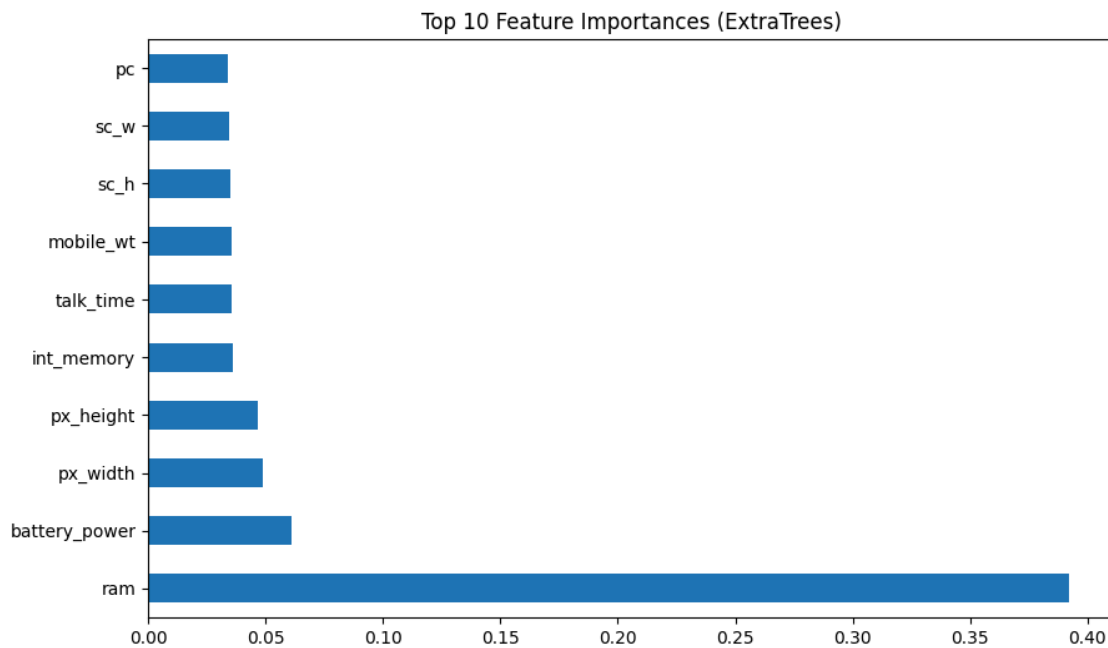
- Initial Accuracy (All 20 Features): 89.00%
 - Final Accuracy (Top 10 Selected Features): 91.75%
-

3. Data Exploration Visualizations

3.1. Feature Correlation Heatmap



3.2. Top 10 Feature Importances



4. Full Source Code

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier, ExtraTreesClassifier
from sklearn.feature_selection import SelectKBest, f_classif
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

1. Load Dataset

```
try:
    df = pd.read_csv('kaggaleproject2/train.csv')
    print("Dataset loaded successfully!")
except FileNotFoundError:
    print("Error: train.csv not found in 'kaggaleproject2' folder.")
    exit()
```

2. Handle Missing Values

```

if df.isnull().values.any():
    df.fillna(df.median(), inplace=True)

# Split Features and Target
X = df.drop('price_range', axis=1)
y = df['price_range']

# Feature Scaling
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

# 3. Grid Search Optimization
param_grid = {
    'n_estimators': [50, 100, 150],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5]
}

rf = RandomForestClassifier(random_state=42)
grid_search = GridSearchCV(estimator=rf, param_grid=param_grid, cv=5, n_jobs=-1)
grid_search.fit(X_train, y_train)

best_rf = grid_search.best_params_

# 4. Feature Selection
# Method 1: Univariate Selection
bestfeatures = SelectKBest(score_func=f_classif, k=10)
fit = bestfeatures.fit(df.drop('price_range', axis=1), y)

# Method 2: Tree-based Feature Importance
model = ExtraTreesClassifier(random_state=42)
model.fit(df.drop('price_range', axis=1), y)
feat_importances = pd.Series(model.feature_importances_, index=X.columns)

# 5. Model Evaluation (Selected Features)
selected_features = feat_importances.nlargest(10).index
X_subset = df[selected_features]
X_train_sub, X_test_sub, y_train_sub, y_test_sub = train_test_split(X_subset, y, test_size=0.2,
random_state=42)

rf_final = RandomForestClassifier(**best_rf, random_state=42)

```

```
rf_final.fit(X_train_sub, y_train_sub)
y_pred = rf_final.predict(X_test_sub)

print(f"Final Accuracy: {accuracy_score(y_test_sub, y_pred)}")
print(classification_report(y_test_sub, y_pred))
```

End of Project Report

Project 1: Data Exploration, Cleaning, and Visualization

Student Name: Ahmed Elnoor Adam Ahmed

Course: Practical Machine Learning and Data Exploration (First Trimester 2026)

Date: 26 April 2026

1. Project Background

This project focuses on the fundamental steps of data preprocessing: handling missing information, identifying and managing outliers, and visualizing key patterns in the data. The Titanic Dataset from Kaggle was used for this analysis.

2. Data Cleaning process

2.1. Handling Missing Values

The dataset contained several missing values which were handled as follows:

- Age: Missing values were filled using the Median age of the passengers.
- Embarked: Missing values were filled with the Mode (most frequent docking port).
- Cabin: Due to the high percentage of missing values, these were labeled as "Unknown" to preserve the record without introducing bias.

2.2. Outlier Management (IQR Method)

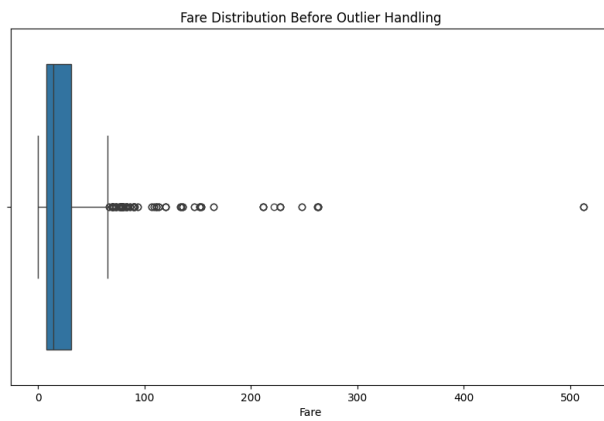
The 'Fare' column exhibited significant outliers. To prevent these from skewing the future model results, the Interquartile Range (IQR) method was applied:

- Upper Bound Identification: Any fare above
 - $Q3 + 1.5 \times IQR$
 - $Q3 + 1.5 \times IQR$ was considered an outlier.
 - Capping: Outliers were capped to the upper and lower bounds rather than being deleted, ensuring we retained the data records.
-

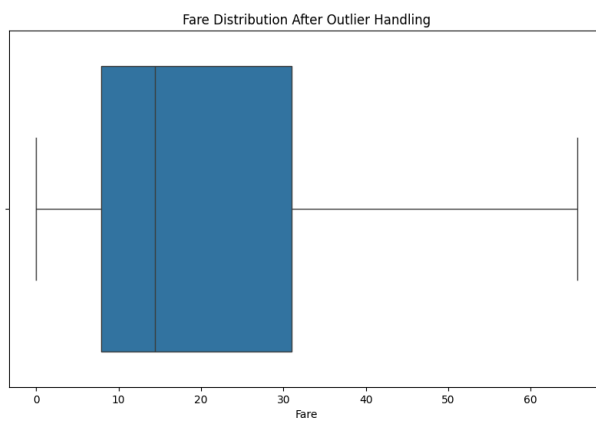
3. Data Visualizations

3.1. Outlier Detection (Before & After)

Before Outlier Handling

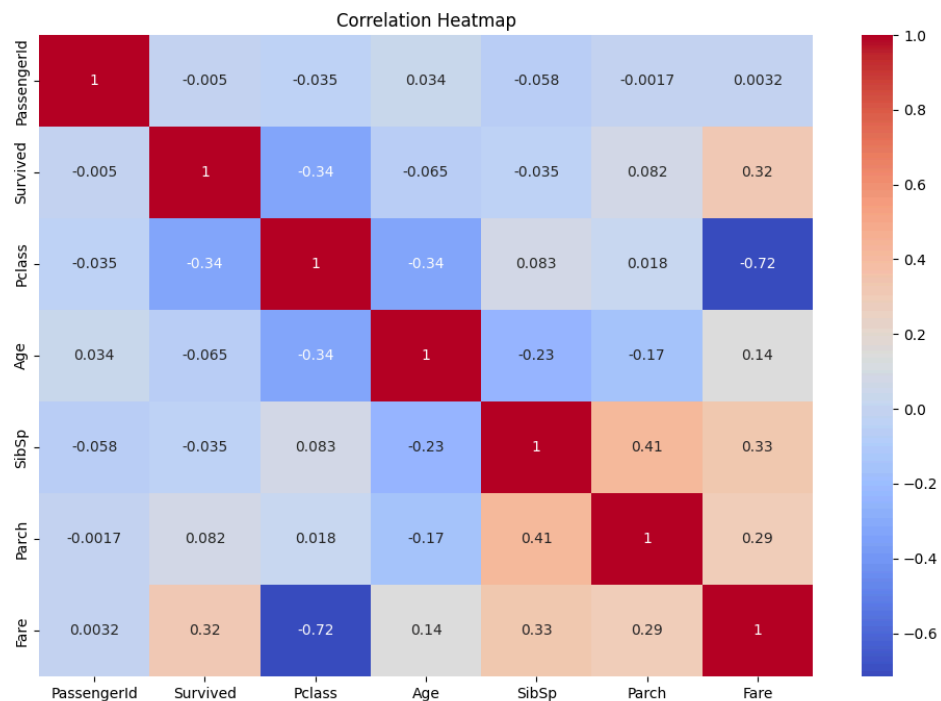


After Outlier Handling



3.2. Distribution and Correlation

- Survival by Class: Visualizes the survival rates across different passenger classes.
- Correlation Heatmap: Identifies relationships between numeric features.



4. Source Code

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

```
# Load Data
```

```
df = pd.read_csv('train.csv')
```

```
# Handling Missing Values
```

```
df['Age'] = df['Age'].fillna(df['Age'].median())
```

```
df['Embarked'] = df['Embarked'].fillna(df['Embarked'].mode()[0])
```

```
df['Cabin'] = df['Cabin'].fillna('Unknown')
```

```
# Handling Outliers (Fare)
```

```
Q1 = df['Fare'].quantile(0.25)
```

```
Q3 = df['Fare'].quantile(0.75)
IQR = Q3 - Q1
upper_bound = Q3 + 1.5 * IQR
df['Fare'] = np.where(df['Fare'] > upper_bound, upper_bound, df['Fare'])

# Visualizations
plt.figure(figsize=(10,6))
sns.boxplot(x=df['Fare']).set_title('Cleaned Fare Boxplot')
plt.savefig('cleaned_fare.png')

plt.figure(figsize=(12,8))
sns.heatmap(df.select_dtypes(include=[np.number]).corr(), annot=True)
plt.savefig('heatmap.png')
```

End of Project Report