Introduction

In this report, we are going to do a critical analysis of a Machine learning project that attempts to predict the price range of different Mobiles. The dataset contains multiple features of mobile phones such as battery power, screen type, number of cores, etc. The project used multiple data analysis techniques, an MLP classifier model, a decision tree-based model, and model evaluation methods. However, the evaluation scores of the model are very poor. We are going to deeply analyze the project and find some critical issues that might have caused the poor performance of the model.

Issues and solutions

Critical Issue 1: Feature selection based on Correlation

In the notebook, correlation values are calculated to understand the correlation between price_range and other features. Correlation can be strongly positive and strongly negative. Negative correlation values don't always indicate weak relationships. The closer the correlation value is to 0, the weaker the relationship is. However, the notebook dropped all the columns with a negative correlation value and left the weakly positive correlated features in the training and testing set. Using unrelated features for the training set creates noise and might result in overfitting or underfitting of data.

Solution

We need to do a through analysis about the correlation matrix. We can observe that clock_speed, m_dep, n_cores, etc have very low correlation values. These features are unrelated to the price range of mobile phones.

Code Sample

```
data = data.drop(columns=['blue', 'clock_speed', 'dual_sim', 'four_g', 'm_dep', 'mobile_wt', 'sc_h',
```

Critical Issue 2: Model Training

The neural network model used in this notebook shows fluctuating accuracy which is visible in the accuracy vs epoch graph. The sudden rise and fall in accuracy indicates overfitting and underfitting of some data. This can cause poor generalization on unseen data which can reduce the test accuracy.

Solution

Trying different learning rates, regularization techniques like dropout and cross validation techniques might improve the situation.

Critical Issue 3: Cross Validation

The notebook does not use cross validation which might cause the model's evaluation to be biased. This can make the model work poorly on unseen data.

Solution

Using K-Fold cross validation can make the model more robust.

Sample Code

```
model = RandomForestClassifier()
scores = cross_val_score(model, X, y, cv=5, scoring='accuracy')
print(f"Cross-validation scores: {scores}")
print(f"Mean accuracy: {scores.mean()}")
```

Non Critical Issue 1: Train Test Split

The train test split method used here is unusual and it has less readability. Usually, the train test split can be done in a more convenient way.

Code Sample

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, stratify=y, random_state=42
```

Non critical issue 2: Code Efficiency

Some operations in the notebook can be done more efficiently. One of the examples are given below.

Inefficient Code

```
y_ex1 = data.copy()['price_range']
```

The above code creates an unnecessary copy of the data frame for a single column.

Efficient Code

```
y_ex1 = data['price_range']
```

Conclusion

The notebook contains multiple wrong approaches which have caused the poor score of the models. Moreover, code readability is poor in this notebook which makes it hard to understand the steps. Python has rich built-in function collections that can be used for more efficient coding. Moreover, the dataset has issues causing the models' fluctuating accuracy. Overall, better data analysis and model building needs to be used here for better performance.