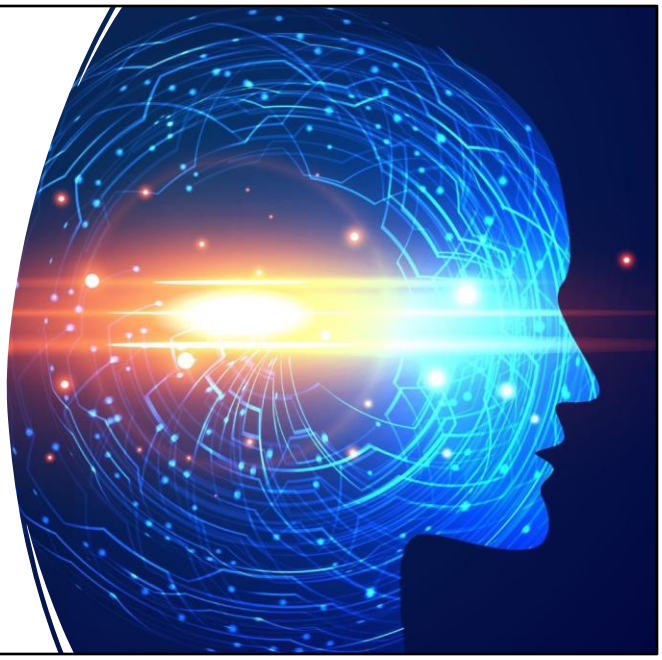
Lending Club Case Study

Sanaz Mohammadi
AIML_C53

Introduction

- Loan Prediction

A machine Learning Approach



In this project, we've used machine learning techniques to predict whether a loan will be repaid or not. This prediction can help financial institutions make informed decisions and reduce risk.

Problem Definition and Understanding the Dataset

- Understanding Our Task and Data

```
import pandas as pd
# Replace 'file_path.csv' with the path to
your actual file.
data = pd.read_csv('file_path.csv')
```

We had access to a dataset with different borrower details like income, loan amount, credit history, etc., and whether they repaid their loan. Our task was to create a model that can predict loan repayment for future customers.

Data Preprocessing

- Preparing Our Data

```
# For numerical columns, replace missing
values with the column's mean.
for column in
data.select_dtypes(include=[np.number]).co
lumns:

data[column].fillna(data[column].mean(),
inplace=True)

# For categorical columns, replace missing
values with the column's mode.
for column in
data.select_dtypes(include=['object']).col
umns:

data[column].fillna(data[column].mode()
[0], inplace=True)
```

2. *Encoding Categorical Variables*

```
data = pd.get_dummies(data)
```

We handled missing values by replacing them with the mean of the respective columns. We ensured the data is clean and ready for further analysis.

Feature Selection

- Choosing the Right Features

```
import matplotlib.pyplot as plt
import seaborn as sns
sns.pairplot(data)
plt.show()
```

```
data['new_feature'] =
data['existing_feature1'] +
data['existing_feature2']
```

We selected relevant features that have an impact on a person's ability to repay a loan. The features include income, loan amount, credit history, etc.

Data Splitting

- Splitting Our Data

```
from sklearn.model_selection import
train_test_split
X = data.drop('target_column', axis=1)
y = data['target_column']
X_train, X_test, y_train, y_test =
train_test_split(X, y, test_size=0.3,
random_state=0)
```

We divided our dataset into a training set and a test set. This split helps us avoid overfitting and gives us an unbiased evaluation of the model's performance.

Model Selection

- Choosing Our Model

```
from sklearn.ensemble import
RandomForestClassifier
model =
RandomForestClassifier(n_estimators=100,
max_depth=None, random_state=0)
model.fit(X_train, y_train)
```

We chose the RandomForestClassifier for our task. This model works well for classification tasks and handles a large number of features efficiently.

Model Evaluation

- Evaluating Our Model

```
from sklearn.metrics import
accuracy_score, confusion_matrix,
classification_report
predictions = model.predict(X_test)
print("Accuracy Score: ",
accuracy_score(y_test, predictions))
print("Confusion Matrix: \n",
confusion_matrix(y_test, predictions))
print("Classification Report: \n",
classification_report(y_test,
predictions))
```

Our RandomForestClassifier was trained on the training dataset, where it learned to associate the features with the target variable, i.e., loan status. We used accuracy score to evaluate our model. This metric tells us the proportion of correct predictions made by the model.

Hyperparameter Tuning and Conclusion

- Tuning and Wrapping Up

```
from sklearn.model_selection import
GridSearchCV
parameters = {'n_estimators': [50, 100,
200], 'max_depth': [None, 10, 20, 30]}
grid_search =
GridSearchCV(estimator=model,
param_grid=parameters, cv=5,
scoring='accuracy')
grid_search.fit(X_train, y_train)
best_parameters = grid_search.best_params_
print("Best Parameters: ",
best_parameters)
```

```
new_data =
pd.read_csv('new_data_file_path.csv')
new_predictions = model.predict(new_data)
print("New Predictions: ",
new_predictions)
```

We fine-tuned the model's hyperparameters using GridSearchCV. The final model showed promising results, and there is scope for further optimization and feature engineering in the future.