Kaggle Seasons #04

QUEEN MARY MACHINE
LEARNING SOCIETY

Encoding - Categorical

- What is Categorical Data?
  - Ordinal (e.g. Shirt sizes: Small < Medium < Large)
  - Nominal (e.g., Gender: Male/Female)
- How to Encode it?
  - Label Encoding (Mapping)
  - One-Hot-Encoding

kaggle

Template: Load & Preprocess

Imports and Installs

- Data Loading

(Optional) Data Enhancement

(Optional) EDA (Explorative Data Analysis)

- Data Preprocessing
  - Data Cleaning

Missing Value Handling

Encoding

(Optional) Feature Engineering

Train-Test Split

Normalisation

kaggle

Template: Model Handling

Model Handling

Model Selection

Model Evaluation

```
# Model training
# Model predictions
# Performance summary

### Recommended: complete the model training and model prediction during cross-validation
```

Model Tuning

Hyperparameter Optimisation

(Optional) Parameter Summary

(Optional) Ensemble Learning

Training

Evaluation

Submission

Data Preprocessing

Prediction

Submission File Creation

kaggle

Sample Submission (Pt. 1)

Imports and Installs

```
[ ] import pandas as pd
    import numpy as np
    from sklearn.preprocessing import OneHotEncoder
    from sklearn.model_selection import train_test_split, StratifiedKFold
    from sklearn.metrics import accuracy_score
    from sklearn.linear_model import LogisticRegression
```

Data Loading

```
[ ] data = pd.read_csv('train.csv')
    test_data = pd.read_csv('test.csv')
```

```
[ ] data = data.drop(columns=['Name', 'id'])
    test_data = test_data.drop(columns=['Name'])
```

Data Preprocessing

Data Cleaning

```
for d in [data, test_data]:
  for column in d.columns:
    if d.isna().sum()[column] > 0:
      mode_value = d[column].dropna().mode()[0]
      d[column].fillna(mode_value, inplace=True)
```

```
<ipython-input-278-d28dd92aa4c8>:5: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.


  d[column].fillna(mode_value, inplace=True)
```

Encoding

```
[ ] nominal_cols = data.select_dtypes(include=['object', 'category']).columns.tolist()
    ohe = OneHotEncoder(sparse_output=False, handle_unknown='ignore')
    data_ohe = pd.DataFrame(ohe.fit_transform(data[nominal_cols]), columns=ohe.get_feature_names_out(nominal_cols), index=data.index)
    test_data_ohe = pd.DataFrame(ohe.transform(test_data[nominal_cols]), columns=ohe.get_feature_names_out(nominal_cols), index=test_data.index)
    data = data.drop(columns=nominal_cols).join(data_ohe)
    test_data = test_data.drop(columns=nominal_cols).join(test_data_ohe)
```

kaggle

Sample Submission (Pt. 2)

Train-Test Split

```
[ ] # Train-test split already performed by Kaggle
    X_train = data.drop(columns=["Depression"])
    y_train = data['Depression']
    X_test = test_data
```

Normalisation

```
[ ] for col in X_train.columns:
      if col not in nominal_cols:
        mu = X_train[col].mean()
        sigma = X_train[col].std()
        X_train[col] = (X_train[col] - mu) / sigma
        X_test[col] = (X_test[col] - mu) / sigma
```

Model Handling

Model Selection

```
[ ] model = LogisticRegression(random_state=42)
```

Model Evaluation

```
[ ] def cross_validate(model, X_train, y_train, cv=5):
      skf = StratifiedKFold(n_splits=cv, shuffle=True, random_state=42)
      accuracy_scores = []

      for train_index, test_index in skf.split(X_train, y_train):
      # Model training
          X_train_fold, X_test_fold = X_train.iloc[train_index], X_train.iloc[test_index]
          y_train_fold, y_test_fold = y_train.iloc[train_index], y_train.iloc[test_index]
          model.fit(X_train_fold, y_train_fold)

      # Model predictions
          y_pred = model.predict(X_test_fold)
          accuracy = accuracy_score(y_test_fold, y_pred)
          accuracy_scores.append(accuracy)

      # Performance summary
      return np.mean(accuracy_scores)

    cross_validate(model, X_train, y_train)
```

```
0.9382871357498223
```

Sample Submission (Pt. 3)

Model Tuning

Hyperparameter Tuning

```
[ ] params = {
        'max_iter': 1000,
        'class_weight': 'balanced'
        }
```

Re-evaluation

```
[ ] model_ht = LogisticRegression(**params)
    cross_validate(model, X_train, y_train)
```

```
0.9382871357498223
```

Submission

```
[ ] # Without hyperparameter tuning
    model.fit(X_train, y_train)
    test_predictions = model.predict(X_test.drop(columns=['id']))
    submission = pd.DataFrame({'id': X_test["id"], 'Depression': test_predictions})
    submission.to_csv('basic_submission.csv', index=False)
```

```
[ ] # With hyperparamter tuning
    model_ht.fit(X_train, y_train)
    test_predictions = model_ht.predict(X_test.drop(columns=['id']))
    submission = pd.DataFrame({'id': X_test["id"], 'Depression': test_predictions})
    submission.to_csv('tuned_submission.csv', index=False)
```

kaggle

Good luck!

QUEEN MARY MACHINE
LEARNING SOCIETY