TASK - 5

Engineer new features and select relevant features for model selection.

1. **Import Libraries and Load Data**

In [2]:
```
import pandas as pd
import numpy as np
from sklearn.datasets import make_classification
X, y = make_classification(
    n_samples=1000,
    n_features=10,
    n_informative=5,
    n_redundant=3,
    n_repeated=0,
    n_classes=2,
    random_state=42
)
feature_names = [f'feature_{i}' for i in range(X.shape[1])]
df = pd.DataFrame(X, columns=feature_names)
df['target'] = y
print(df.head())
```

```
   feature_0  feature_1  feature_2  feature_3  feature_4  feature_5  \
0   0.832566   3.984819   1.371106  -0.566705  -0.942890   2.277278
1  -1.369035   0.231141   1.802292  -0.032047  -1.136737   0.651840
2   2.830853  -1.946436  -1.881707  -0.161955   1.611247   0.064322
3  -2.775746  -1.035596   1.387249   0.061883   1.157426  -1.201067
4  -0.908299   2.494992   1.265136  -0.981326  -0.222445   0.275819

   feature_6  feature_7  feature_8  feature_9  target
0  -0.599821   3.790909   2.856752  -1.184140       1
1   0.033525   3.660358  -1.237293   1.790979       1
2   0.438413  -2.520883  -1.355660   1.555376       0
3  -0.927553  -2.798974  -3.356626  -0.318152       0
4  -1.404817   1.276871   2.433687  -1.909264       1
```

2. **Feature Engineering**

In [3]:
```
df['feature_0_1_interaction'] = df['feature_0'] * df['feature_1']
df['hour'] = np.random.randint(0, 24, size=len(df))
df['hour_sin'] = np.sin(2 * np.pi * df['hour'] / 24)
df['hour_cos'] = np.cos(2 * np.pi * df['hour'] / 24)

df = df.drop('hour', axis=1)

print(df.head())
```

```
   feature_0  feature_1  feature_2  feature_3  feature_4  feature_5  \
0   0.832566   3.984819   1.371106  -0.566705  -0.942890   2.277278
1  -1.369035   0.231141   1.802292  -0.032047  -1.136737   0.651840
2   2.830853  -1.946436  -1.881707  -0.161955   1.611247   0.064322
3  -2.775746  -1.035596   1.387249   0.061883   1.157426  -1.201067
4  -0.908299   2.494992   1.265136  -0.981326  -0.222445   0.275819

   feature_6  feature_7  feature_8  feature_9  target  \
0  -0.599821   3.790909   2.856752  -1.184140       1
1   0.033525   3.660358  -1.237293   1.790979       1
2   0.438413  -2.520883  -1.355660   1.555376       0
3  -0.927553  -2.798974  -3.356626  -0.318152       0
4  -1.404817   1.276871   2.433687  -1.909264       1

   feature_0_1_interaction  hour_sin  hour_cos
0                 3.317625 -0.965926 -0.258819
1                -0.316441  0.258819  0.965926
2                -5.510076  0.000000  1.000000
3                 2.874551  0.500000 -0.866025
4                -2.266198  0.866025 -0.500000
```

3. Feature Selection

In [12]:
```
df_filtered = df.loc[:, df.nunique() > 1]
print(df_filtered.head())
```

```
   feature_0  feature_1  feature_2  feature_3  feature_4  feature_5  \
0   0.832566   3.984819   1.371106  -0.566705  -0.942890   2.277278
1  -1.369035   0.231141   1.802292  -0.032047  -1.136737   0.651840
2   2.830853  -1.946436  -1.881707  -0.161955   1.611247   0.064322
3  -2.775746  -1.035596   1.387249   0.061883   1.157426  -1.201067
4  -0.908299   2.494992   1.265136  -0.981326  -0.222445   0.275819

   feature_6  feature_7  feature_8  feature_9  target  \
0  -0.599821   3.790909   2.856752  -1.184140       1
1   0.033525   3.660358  -1.237293   1.790979       1
2   0.438413  -2.520883  -1.355660   1.555376       0
3  -0.927553  -2.798974  -3.356626  -0.318152       0
4  -1.404817   1.276871   2.433687  -1.909264       1

   feature_0_1_interaction  hour_sin  hour_cos
0                 3.317625 -0.965926 -0.258819
1                -0.316441  0.258819  0.965926
2                -5.510076  0.000000  1.000000
3                 2.874551  0.500000 -0.866025
4                -2.266198  0.866025 -0.500000
```

In [13]:
```
corr_matrix = df_filtered.corr().abs()
upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype(bool))
to_drop = [column for column in upper.columns if any(upper[column] > 0.9)]
df_filtered = df_filtered.drop(to_drop, axis=1)
print(df_filtered.head())
```

```
   feature_0  feature_1  feature_2  feature_3  feature_4  feature_5  \
0   0.832566   3.984819   1.371106  -0.566705  -0.942890   2.277278
1  -1.369035   0.231141   1.802292  -0.032047  -1.136737   0.651840
2   2.830853  -1.946436  -1.881707  -0.161955   1.611247   0.064322
3  -2.775746  -1.035596   1.387249   0.061883   1.157426  -1.201067
4  -0.908299   2.494992   1.265136  -0.981326  -0.222445   0.275819

   feature_6  feature_7  feature_8  feature_9  target  \
0  -0.599821   3.790909   2.856752  -1.184140       1
1   0.033525   3.660358  -1.237293   1.790979       1
2   0.438413  -2.520883  -1.355660   1.555376       0
3  -0.927553  -2.798974  -3.356626  -0.318152       0
4  -1.404817   1.276871   2.433687  -1.909264       1

   feature_0_1_interaction  hour_sin  hour_cos
0                 3.317625 -0.965926 -0.258819
1                -0.316441  0.258819  0.965926
2                -5.510076  0.000000  1.000000
3                 2.874551  0.500000 -0.866025
4                -2.266198  0.866025 -0.500000
```

In [15]:
```
from sklearn.feature_selection import SelectKBest, f_classif
X_filtered = df_filtered.drop('target', axis=1)
y_filtered = df_filtered['target']
selector = SelectKBest(f_classif, k=5)
X_selected = selector.fit_transform(X_filtered, y_filtered)
selected_features = X_filtered.columns[selector.get_support()]
print("Selected features:", selected_features)
```

```
Selected features: Index(['feature_1', 'feature_4', 'feature_5', 'feature_8', 'feature_9'], dtype='object')
```

In [16]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.feature_selection import RFE
model = LogisticRegression()
rfe = RFE(model, n_features_to_select=5)
X_rfe = rfe.fit_transform(X_filtered, y_filtered)
selected_features_rfe = X_filtered.columns[rfe.support_]
print("Selected features via RFE:", selected_features_rfe)
```

```
Selected features via RFE: Index(['feature_1', 'feature_2', 'feature_4', 'feature_8', 'feature_9'], dtype='object')
```

4. Evaluating Feature Importance

In [17]:

```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier()
model.fit(X_filtered, y_filtered)
importances = model.feature_importances_
indices = np.argsort(importances)[::-1]
for f in range(X_filtered.shape[1]):
    print(f"{f + 1}. Feature {X_filtered.columns[indices[f]]} ({importances[indices[f]]})")
```

```
1. Feature feature_9 (0.22590060967237963)
2. Feature feature_4 (0.15457053644989083)
3. Feature feature_1 (0.10973849849279096)
4. Feature feature_8 (0.10677224737617406)
5. Feature feature_5 (0.0980771211975233)
6. Feature feature_7 (0.0729175074328932)
7. Feature feature_0_1_interaction (0.06189460903369599)
8. Feature feature_2 (0.05490334510748982)
9. Feature feature_0 (0.050536394582553684)
10. Feature feature_3 (0.02355646697005031)
11. Feature feature_6 (0.02017057835397781)
12. Feature hour_sin (0.010503391775039044)
13. Feature hour_cos (0.010458693555541203)
```