In [1]:
```
# 1233745983
```

In [2]:
```
# Q1: PREPROCESSING
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.model_selection import StratifiedKFold, GridSearchCV, cross_val_score
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder, FunctionTransformer
from sklearn.impute import SimpleImputer
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.ensemble import RandomForestClassifier

pd.set_option('display.max_columns', 200)
plt.rcParams['figure.figsize'] = (12, 8)

TRAIN_CSV = "/Users/qayyumkhan/Downloads/train.csv"
df = pd.read_csv(TRAIN_CSV)

def extract_title(name: str) -> str:
    title = name.split(',')[1].split('.')[0].strip()
    title = title.replace('Mlle', 'Miss').replace('Ms', 'Miss').replace('Mme', 'Mrs')
    common = {'Mr','Miss','Mrs','Master','Dr','Rev','Col','Lady','Sir','Major','Countess','Capt','D
    return title if title in common else "Rare"

def add_title_column(X: pd.DataFrame) -> pd.DataFrame:
    X = X.copy()
    X['Title'] = X['Name'].apply(extract_title)
    return X

# Columns we will use for modeling
target_col = 'Survived'
base_features = ['Pclass','Name','Sex','Age','SibSp','Parch','Fare','Embarked']
numeric_features = ['Age','SibSp','Parch','Fare']
categorical_features = ['Pclass','Sex','Embarked','Title']
```

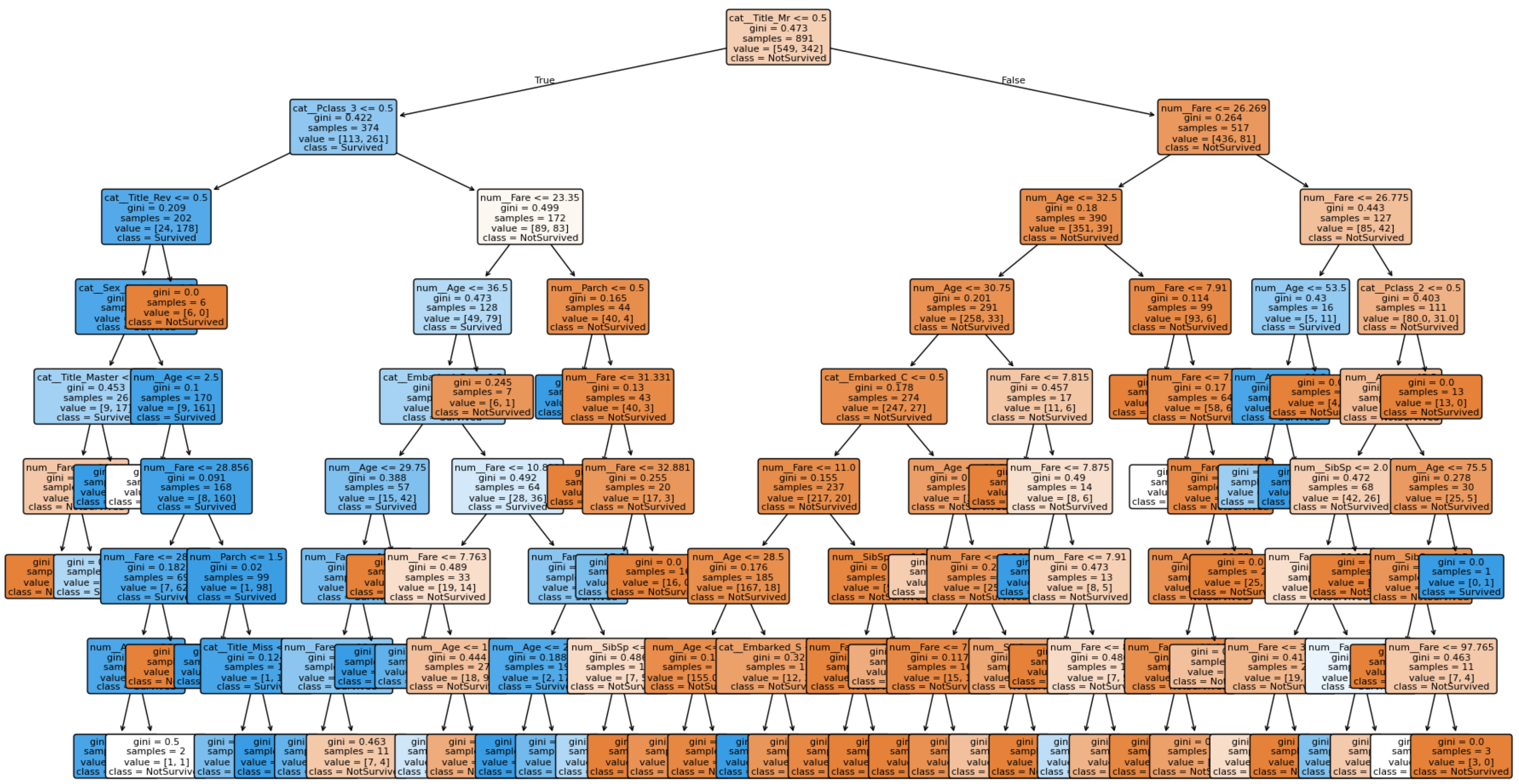
```
def select_needed_columns(X: pd.DataFrame) -> pd.DataFrame:
    cols = list(dict.fromkeys(base_features + ['Title']))
    return X[cols]

# Pipelines
numeric_tf = Pipeline([('imputer', SimpleImputer(strategy='median'))])
categorical_tf = Pipeline([('imputer', SimpleImputer(strategy='most_frequent')),
                           ('onehot',  OneHotEncoder(handle_unknown='ignore'))])

preprocess = Pipeline([
    ('add_title', FunctionTransformer(add_title_column, validate=False)),
    ('select',    FunctionTransformer(select_needed_columns, validate=False)),
    ('colxf',     ColumnTransformer([
                    ('num', numeric_tf, numeric_features),
                    ('cat', categorical_tf, categorical_features)
                  ], remainder='drop'))
])

# Splitting features
X = df.drop(columns=[target_col])
y = df[target_col].astype(int)

print("Shape:", X.shape, " Target distribution:\n", y.value_counts(normalize=True).round(3))
```

```
Shape: (891, 11)  Target distribution:
 Survived
0    0.616
1    0.384
Name: proportion, dtype: float64
```

In [3]:
```
# Q2: TUNE + PLOT DECISION TREE
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

dt_pipe = Pipeline([
    ('preprocess', preprocess),
    ('model', DecisionTreeClassifier(random_state=42))
])

param_grid_dt = {
    'model__criterion': ['gini', 'entropy', 'log_loss'],
    'model__max_depth': [3, 4, 5, 6, 8, None],
    'model__min_samples_split': [2, 5, 10],
    'model__min_samples_leaf': [1, 2, 4]
}

gs_dt = GridSearchCV(dt_pipe, param_grid_dt, cv=cv, scoring='accuracy', n_jobs=-1, verbose=0)
gs_dt.fit(X, y)

best_dt = gs_dt.best_estimator_
print("Best DT params:", gs_dt.best_params_)
print("Best DT CV accuracy:", round(gs_dt.best_score_*100, 2), "%")

pre_only = best_dt.named_steps['preprocess'].fit(X, y)
feat_names = pre_only.named_steps['colxf'].get_feature_names_out()

tree_model = best_dt.named_steps['model']
plt.figure(figsize=(18,10))
plot_tree(tree_model, feature_names=feat_names,
          class_names=['NotSurvived','Survived'],
          filled=True, rounded=True, fontsize=8)
plt.title("Decision Tree — Titanic (fine-tuned)")
plt.tight_layout()
plt.show()
```

```
Best DT params: {'model__criterion': 'gini', 'model__max_depth': 8, 'model__min_samples_leaf': 1, '
model__min_samples_split': 10}
Best DT CV accuracy: 83.39 %
```

Decision Tree – Titanic (fine-tuned)



```
In [4]: # Q3: 5-FOLD CV (DECISION TREE)
        cv_scores_dt = cross_val_score(best_dt, X, y, cv=cv, scoring='accuracy', n_jobs=-1)
        print("Decision Tree 5-fold CV mean accuracy:", round(cv_scores_dt.mean()*100, 2), "%")
        print("Decision Tree 5-fold CV std:", round(cv_scores_dt.std()*100, 2), "%")
        cv_scores_dt
```

```
Decision Tree 5-fold CV mean accuracy: 83.39 %
Decision Tree 5-fold CV std: 2.55 %
```

```
Out[4]: array([0.83798883, 0.87078652, 0.79775281, 0.84831461, 0.81460674])
```

```
In [5]: # Q4: 5-FOLD CV (RANDOM FOREST)
        rf_pipe = Pipeline([
            ('preprocess', preprocess),
            ('model', RandomForestClassifier(random_state=42))
        ])

        param_grid_rf = {
            'model__n_estimators': [100, 200, 300],
            'model__max_depth': [None, 5, 8, 12],
            'model__min_samples_leaf': [1, 2, 4],
            'model__max_features': ['sqrt', 1.0]
        }

        gs_rf = GridSearchCV(rf_pipe, param_grid_rf, cv=cv, scoring='accuracy', n_jobs=-1, verbose=0)
        gs_rf.fit(X, y)

        best_rf = gs_rf.best_estimator_
        print("Best RF params:", gs_rf.best_params_)
        print("Best RF CV accuracy:", round(gs_rf.best_score_*100, 2), "%")

        cv_scores_rf = cross_val_score(best_rf, X, y, cv=cv, scoring='accuracy', n_jobs=-1)
        print("Random Forest 5-fold CV mean accuracy:", round(cv_scores_rf.mean()*100, 2), "%")
        print("Random Forest 5-fold CV std:", round(cv_scores_rf.std()*100, 2), "%")
        cv_scores_rf
```

```
Best RF params: {'model__max_depth': 8, 'model__max_features': 1.0, 'model__min_samples_leaf': 1, '
model__n_estimators': 300}
Best RF CV accuracy: 84.73 %
Random Forest 5-fold CV mean accuracy: 84.73 %
Random Forest 5-fold CV std: 1.71 %
```

Out[5]:
```
array([0.87709497, 0.85393258, 0.83146067, 0.84269663, 0.83146067])
```

In [6]:
```
# Q5: COMPARISON + CONCLUSION
summary = pd.DataFrame({
    'Model': ['Decision Tree (tuned)', 'Random Forest (tuned)'],
    'CV Mean Accuracy (%)': [round(cv_scores_dt.mean()*100, 2),
                             round(cv_scores_rf.mean()*100, 2)],
    'CV Std (%)': [round(cv_scores_dt.std()*100, 2),
                   round(cv_scores_rf.std()*100, 2)]
}).sort_values('CV Mean Accuracy (%)', ascending=False).reset_index(drop=True)
display(summary)
```

```
dt_mean, rf_mean = cv_scores_dt.mean(), cv_scores_rf.mean()
if rf_mean > dt_mean + 1e-6:
    print("Conclusion: Random Forest outperforms the Decision Tree on average across folds "
          "due to variance reduction via ensembling (lower variance, better generalization).")
elif dt_mean > rf_mean + 1e-6:
    print("Conclusion: Decision Tree slightly outperforms Random Forest here, though RF is usually
          "more robust; this feature set favored a simpler tree.")
else:
    print("Conclusion: Both models perform similarly. In practice, Random Forest is typically prefe
          "for stability/generalization; a single Decision Tree is easier to interpret.")
```

	Model	CV Mean Accuracy (%)	CV Std (%)
0	Random Forest (tuned)	84.73	1.71
1	Decision Tree (tuned)	83.39	2.55

```
Conclusion: Random Forest outperforms the Decision Tree on average across folds due to variance red
uction via ensembling (lower variance, better generalization).
```