

```

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
import matplotlib.pyplot as plt

titanic_data = pd.read_csv("titanic.csv")

# Handling missing values
titanic_data.fillna(method='ffill', inplace=True)

# Encoding categorical variables
titanic_data = pd.get_dummies(titanic_data, columns=['Sex', 'Embarked'], drop_first=True)

# Selecting features and target variable
X = titanic_data.drop(['Survived', 'Name', 'Ticket', 'Cabin'], axis=1)
y = titanic_data['Survived']

# Splitting the dataset into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Logistic Regression
log_reg = LogisticRegression()
log_reg.fit(X_train, y_train)

# Random Forest Classifier
rf_classifier = RandomForestClassifier()
rf_classifier.fit(X_train, y_train)

# Random Forest Classifier
RandomForestClassifier()

# Logistic Regression
log_reg_pred = log_reg.predict(X_test)
print("Logistic Regression Accuracy:", accuracy_score(y_test, log_reg_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, log_reg_pred))
print("Classification Report:\n", classification_report(y_test, log_reg_pred))

# Random Forest Classifier
rf_pred = rf_classifier.predict(X_test)
print("Random Forest Classifier Accuracy:", accuracy_score(y_test, rf_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, rf_pred))
print("Classification Report:\n", classification_report(y_test, rf_pred))

```

```

Logistic Regression Accuracy: 1.0
Confusion Matrix:
[[1]]
Classification Report:

```

	precision	recall	f1-score	support
1	1.00	1.00	1.00	1
accuracy			1.00	1
macro avg	1.00	1.00	1.00	1
weighted avg	1.00	1.00	1.00	1

```

Random Forest Classifier Accuracy: 1.0
Confusion Matrix:
[[1]]
Classification Report:

```

	precision	recall	f1-score	support
1	1.00	1.00	1.00	1
accuracy			1.00	1
macro avg	1.00	1.00	1.00	1
weighted avg	1.00	1.00	1.00	1

```
# Feature importance for Random Forest Classifier
feat_importances = pd.Series(rf_classifier.feature_importances_, index=X.columns)
feat_importances.nlargest(10).plot(kind='barh')
plt.title('Feature Importance - Random Forest Classifier')
plt.show()
```

