Importing Libraries

```
import pandas as pd
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from scipy import stats
import warnings
from sklearn.model_selection import GridSearchCV
from sklearn.exceptions import FitFailedWarning
```

Importing Data

```
data = pd.read_csv('wineq.csv')
X = data.iloc[:, :-1].values
y = data.iloc[:, -1].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0
```

Algorithm Selection

```
models = [
    ('Random Forest', RandomForestClassifier()),
    ('SVM', SVC()),
    ('Logistic Regression', LogisticRegression()),
    ('Decision Tree', DecisionTreeClassifier())
]
```

Cross-Validation and Model Evaluation

```
from sklearn.exceptions import ConvergenceWarning
warnings.filterwarnings("ignore", category=ConvergenceWarning)
results = {}
for name, model in models:
    scores = cross_val_score(model, X_train, y_train, cv=5, scoring='accuracy')
    results[name] = scores
```

Statistical Tests

Statistical Tests

```
comparisons = []
for i, (name1, _) in enumerate(models):
    for j, (name2, _) in enumerate(models):
        if i < j:
            p_value = stats.ttest_rel(results[name1], results[name2]).pvalue
            comparisons.append((name1, name2, p_value))
```

Model Selection

```
alpha = 0.01  # Set your significance level
selected_algorithms = [name for name, _ in models if all(p >= alpha for _, _, p in comp
```

Final Selection

```
final_scores = {}
for name, model in models:
    model.fit(X_train, y_train)
    accuracy = model.score(X_test, y_test)
    final_scores[name] = accuracy

best_model = max(final_scores, key=final_scores.get)
for name, accuracy in final_scores.items():
    print(f"{name} Accuracy: {accuracy}")
print("Selected Algorithms:", selected_algorithms)
print("Best Model:", best_model, final_scores[best_model])
```

```
Random Forest Accuracy: 0.715625
SVM Accuracy: 0.503125
Logistic Regression Accuracy: 0.63125
Decision Tree Accuracy: 0.646875
Selected Algorithms: ['Random Forest']
Best Model: Random Forest 0.715625
```

Hyperparameter Optimization

```
warnings.filterwarnings("ignore", category=FitFailedWarning)
# Hyperparameter Optimization for Logistic Regression
logreg_param_grid = {
    'C': [0.1, 0.5, 1, 5, 10],
    'penalty': ['15', 'l1', 'l2'],
    'solver': ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga']
}
```

```
}

logreg_optimized = GridSearchCV(LogisticRegression(max_iter=100), param_grid=logreg_par
logreg_optimized.fit(X_train, y_train)
best_logreg_model = logreg_optimized.best_estimator_
logreg_accuracy = best_logreg_model.score(X_test, y_test)

# Hyperparameter Optimization for Decision Tree
dt_param_grid = {
    'max_depth': [None, 10, 20, 30, 50,100],
    'min_samples_split': [2, 5, 10, 15],
    'min_samples_leaf': [1, 2, 4, 7]
}

dt_optimized = GridSearchCV(DecisionTreeClassifier(), param_grid=dt_param_grid, cv=5, s
dt_optimized.fit(X_train, y_train)
best_dt_model = dt_optimized.best_estimator_
dt_accuracy = best_dt_model.score(X_test, y_test)
# Hyperparameter Optimization for Random Forest
rf_param_grid = {
    'n_estimators': [10, 50, 100, 200],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

rf_optimized = GridSearchCV(RandomForestClassifier(), param_grid=rf_param_grid, cv=5, s
rf_optimized.fit(X_train, y_train)
best_rf_model = rf_optimized.best_estimator_
rf_accuracy = best_rf_model.score(X_test, y_test)
# Hyperparameter Optimization for SVM
svm_param_grid = {
    'C': [ 0.1, 0.8, 2, 10],
    'kernel': ['linear', 'rbf', 'poly', 'sigmoid']
}

svm_optimized = GridSearchCV(SVC(), param_grid=svm_param_grid, cv=5, scoring='accuracy'
svm_optimized.fit(X_train, y_train)
best_svm_model = svm_optimized.best_estimator_
svm_accuracy = best_svm_model.score(X_test, y_test)
```

Final Selection after HPO

```
final_scores = {
    'Random Forest': rf_accuracy,
    'SVM': svm_accuracy,
    'Logistic Regression': logreg_accuracy,
    'Decision Tree': dt_accuracy
}
```

```python
best_model = max(final_scores, key=final_scores.get)

for name, accuracy in final_scores.items():
    print(f"{name} Accuracy: {accuracy}")

print("Selected Algorithms:", selected_algorithms)
print("Best Model:", best_model, final_scores[best_model])
```

```
Random Forest Accuracy: 0.73125
SVM Accuracy: 0.6375
Logistic Regression Accuracy: 0.625
Decision Tree Accuracy: 0.6625
Selected Algorithms: ['Random Forest']
Best Model: Random Forest 0.73125
```