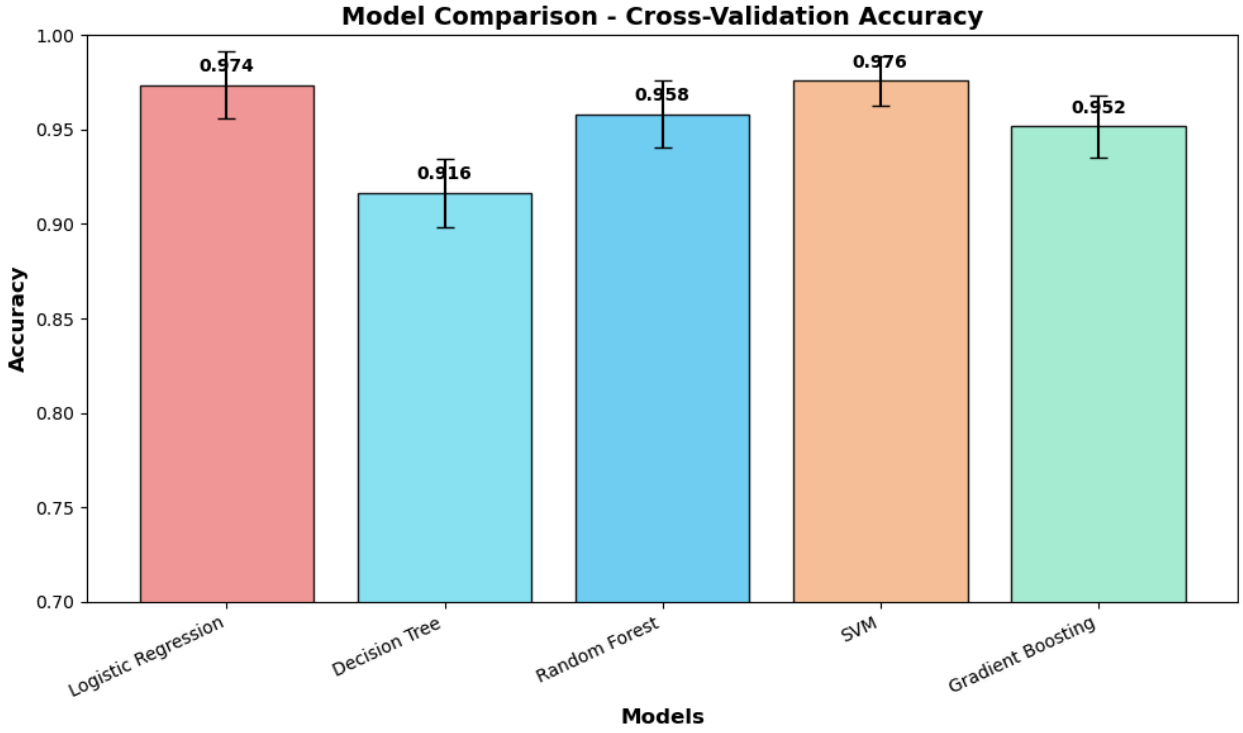
In [1]:
```
#PRACTICAL 9 – Model Selection & Hyperparameter Tuning
```

In [3]:
```
#Name: Ritika R. Junekar
#Sub: PD
#Roll_No:29
```

In [5]:
```
# PRACTICAL 9: Model comparison + GridSearchC

import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import cross_val_score, train_test_split, GridSea
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifie
from sklearn.svm import SVC
from sklearn.datasets import load_breast_cancer
from sklearn.preprocessing import StandardScaler
import warnings
from sklearn.exceptions import ConvergenceWarning

warnings.filterwarnings("ignore", category=ConvergenceWarning)

# Load dataset
data = load_breast_cancer()
X, y = data.data, data.target

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# Scaling (important for LR & SVM)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Define models
models = {
    "Logistic Regression": LogisticRegression(max_iter=5000, random_state=42),
    "Decision Tree": DecisionTreeClassifier(random_state=42),
    "Random Forest": RandomForestClassifier(random_state=42),
    "SVM": SVC(random_state=42),
    "Gradient Boosting": GradientBoostingClassifier(random_state=42),
}

# 5-fold cross-validation
cv_results = {}
print("Model Comparison (5-Fold Cross-Validation):\n")
for name, model in models.items():
    scores = cross_val_score(model, X_train, y_train, cv=5)
    cv_results[name] = (scores.mean(), scores.std())
    print(f"{name} - {scores.mean():.4f} (+/- {scores.std():.4f})")
```

```python
# Grid Search on Random Forest
print("\nGrid Search on Random Forest...")
param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [None, 10, 20]
}
grid = GridSearchCV(
    RandomForestClassifier(random_state=42),
    param_grid,
    cv=5
)
grid.fit(X_train, y_train)

print(f"Best Params: {grid.best_params_}")
print(f"CV Score: {grid.best_score_:.4f}")
print(f"Test Score: {grid.score(X_test, y_test):.4f}")

# Bar plot of CV accuracy
model_names = list(cv_results.keys())
means = [cv_results[m][0] for m in model_names]
stds = [cv_results[m][1] for m in model_names]

colors = ['#F28E8E', '#7BDFF2', '#5EC9F2', '#F8B88B', '#9EEBCF']

plt.figure(figsize=(10, 6))
bars = plt.bar(model_names, means, yerr=stds, capsize=5,
               color=colors, alpha=0.9, edgecolor='black')

for bar, mean in zip(bars, means):
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2,
             yval + 0.005,
             f"{mean:.3f}",
             ha='center', va='bottom',
             fontsize=10, fontweight='bold')

plt.ylim(0.7, 1.0)
plt.ylabel("Accuracy", fontsize=12, fontweight='bold')
plt.xlabel("Models", fontsize=12, fontweight='bold')
plt.title("Model Comparison - Cross-Validation Accuracy",
          fontsize=14, fontweight='bold')
plt.xticks(rotation=25, ha='right')
plt.tight_layout()
plt.show()
```

```
Model Comparison (5-Fold Cross-Validation):

Logistic Regression - 0.9736 (+/- 0.0179)
Decision Tree - 0.9165 (+/- 0.0179)
Random Forest - 0.9582 (+/- 0.0176)
SVM - 0.9758 (+/- 0.0128)
Gradient Boosting - 0.9516 (+/- 0.0164)

Grid Search on Random Forest...
Best Params: {'max_depth': None, 'n_estimators': 200}
CV Score: 0.9626
Test Score: 0.9649
```



In [ ]: