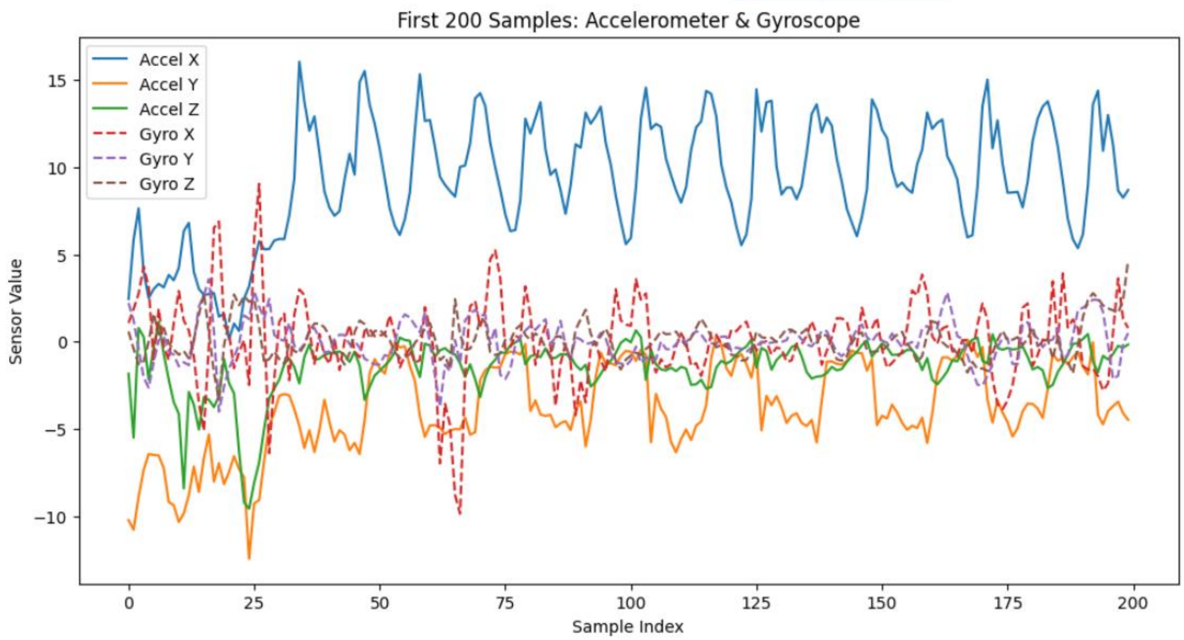
```python
# --- Line Plot: First 200 Samples ---
plt.figure(figsize=(12,6))
plt.plot(df_watch_accel["x"][:200], label="Accel X")
plt.plot(df_watch_accel["y"][:200], label="Accel Y")
plt.plot(df_watch_accel["z"][:200], label="Accel Z")
plt.plot(df_watch_gyro["x"][:200], label="Gyro X", linestyle='--')
plt.plot(df_watch_gyro["y"][:200], label="Gyro Y", linestyle='--')
plt.plot(df_watch_gyro["z"][:200], label="Gyro Z", linestyle='--')
plt.legend()
plt.title("First 200 Samples: Accelerometer & Gyroscope")
plt.xlabel("Sample Index")
plt.ylabel("Sensor Value")
plt.show()
```




```python
# --- Baseline Models: Train & Evaluate ---
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, f1_score
from sklearn.model_selection import train_test_split

# Train/test split
X_train, X_test, y_train, y_test = train_test_split(
    features_combined, labels_trim, test_size=0.2, random_state=42, stratify=labels_trim
)

# Define baseline models
models = {
    "Random Forest": RandomForestClassifier(n_estimators=50, random_state=42),
    "KNN": KNeighborsClassifier(n_neighbors=5),
    "SVM": SVC(kernel='linear', random_state=42)
}

results = []
```

```python
# Train & evaluate
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred, average='weighted')
    results.append((name, acc, f1))
    print(f"{name}: Accuracy={acc:.3f}, F1={f1:.3f}")

# Save results table
import pandas as pd
results_df = pd.DataFrame(results, columns=["Model", "Accuracy", "F1"])
results_df
```

```
Random Forest: Accuracy=0.904, F1=0.903
KNN: Accuracy=0.835, F1=0.833
SVM: Accuracy=0.842, F1=0.839
```

	Model	Accuracy	F1
0	Random Forest	0.903846	0.903174
1	KNN	0.834615	0.832643
2	SVM	0.842308	0.839313

```python
# --- Estimate Model Sizes ---
import joblib
import os

for name, model in models.items():
    filename = f"{name.replace(' ','_')}.joblib"
    joblib.dump(model, filename)
    size_mb = os.path.getsize(filename) / (1024*1024)
    print(f"{name} size: {size_mb:.2f} MB")
```

```
Random Forest size: 2.51 MB
KNN size: 0.20 MB
SVM size: 0.21 MB
```

```python
# --- Simple Compression: Random Forest with fewer trees ---
rf_small = RandomForestClassifier(n_estimators=10, random_state=42)
rf_small.fit(X_train, y_train)
y_pred_small = rf_small.predict(X_test)

acc_small = accuracy_score(y_test, y_pred_small)
f1_small = f1_score(y_test, y_pred_small, average='weighted')

print(f"Compressed RF: Accuracy={acc_small:.3f}, F1={f1_small:.3f}")

# Save compressed model
joblib.dump(rf_small, "RandomForest_Compressed.joblib")
```

```
Compressed RF: Accuracy=0.888, F1=0.888
['RandomForest_Compressed.joblib']
```

```python
# --- Prepare Phone Features & Labels ---
import os
import pandas as pd
import numpy as np

# Paths to phone accel & gyro folders
phone_accel_dir = "/kaggle/input/activity-recognition-dataset/phone_csv/phone_csv/accel"
phone_gyro_dir  = "/kaggle/input/activity-recognition-dataset/phone_csv/phone_csv/gyro"

# Pick the first CSV in each folder
phone_accel_file = os.path.join(phone_accel_dir, os.listdir(phone_accel_dir)[0])
phone_gyro_file  = os.path.join(phone_gyro_dir, os.listdir(phone_gyro_dir)[0])

df_phone_accel = pd.read_csv(phone_accel_file)
df_phone_gyro  = pd.read_csv(phone_gyro_file)

# Sliding window feature extraction
def sliding_window_features(df, axes, window_size=50, step_size=50):
    feats = []
    for start in range(0, len(df) - window_size + 1, step_size):
        window = df[axes].iloc[start:start+window_size]
        feat = np.concatenate([
            window.mean().values,
            window.std().values,
            window.min().values,
            window.max().values
```

```python
axes = ['x','y','z']
feat_accel_phone = sliding_window_features(df_phone_accel, axes)
feat_gyro_phone  = sliding_window_features(df_phone_gyro, axes)

# Combine accel + gyro
min_len_phone = min(feat_accel_phone.shape[0], feat_gyro_phone.shape[0])
features_phone_combined = np.concatenate([feat_accel_phone[:min_len_phone], feat_gyro_phone[:min_len_pho

# Create labels
def sliding_window_labels(df, label_col, window_size=50, step_size=50):
    labels = []
    for start in range(0, len(df) - window_size + 1, step_size):
        window = df[label_col].iloc[start:start+window_size]
        labels.append(window.mode()[0])
    return np.array(labels)

labels_phone = sliding_window_labels(df_phone_accel, 'activity_code')
labels_phone_trim = labels_phone[:min_len_phone]

print("Phone features shape:", features_phone_combined.shape)
print("Number of unique phone labels:", len(np.unique(labels_phone_trim)))
```

```
Phone features shape: (1286, 24)
Number of unique phone labels: 18
```

```python
# --- Cell 4: Cross-device Training ---

# Make sure phone features & labels are prepared
# If not already done, run the phone features extraction cell first

from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, f1_score

# Train on Watch, test on Phone
rf_cross = RandomForestClassifier(n_estimators=50, random_state=42)
rf_cross.fit(features_combined, labels_trim)
y_pred_cross = rf_cross.predict(features_phone_combined)

acc_cross = accuracy_score(labels_phone_trim, y_pred_cross)
f1_cross = f1_score(labels_phone_trim, y_pred_cross, average='weighted')
print(f"Watch → Phone: Accuracy={acc_cross:.3f}, F1={f1_cross:.3f}")

# Train on Phone, test on Watch
rf_cross2 = RandomForestClassifier(n_estimators=50, random_state=42)
rf_cross2.fit(features_phone_combined, labels_phone_trim)
y_pred_cross2 = rf_cross2.predict(features_combined)

acc_cross2 = accuracy_score(labels_trim, y_pred_cross2)
f1_cross2 = f1_score(labels_trim, y_pred_cross2, average='weighted')
print(f"Phone → Watch: Accuracy={acc_cross2:.3f}, F1={f1_cross2:.3f}")
```

```
Watch → Phone: Accuracy=0.156, F1=0.099
Phone → Watch: Accuracy=0.089, F1=0.066
```

+ Code + Markdown

```python
# --- Cell 5: Summary Table of Results ---
import pandas as pd

# Collect baseline model metrics
baseline_acc = [results_df.loc[0,"Accuracy"], results_df.loc[1,"Accuracy"], results_df.loc[2,"Accuracy"]
baseline_f1  = [results_df.loc[0,"F1"], results_df.loc[1,"F1"], results_df.loc[2,"F1"]]

# Add compressed RF and cross-device metrics
all_acc = baseline_acc + [acc_small, acc_cross, acc_cross2]
all_f1  = baseline_f1  + [f1_small, f1_cross, f1_cross2]

all_models = ["RF Baseline", "KNN Baseline", "SVM Baseline", "RF Compressed", "Watch→Phone", "Phone→Watc
```


```python
all_models = ["RF Baseline", "KNN Baseline", "SVM Baseline", "RF Compressed", "Watch→Phone", "Phone→Watc
```


```python
summary_df = pd.DataFrame({
    "Model": all_models,
    "Accuracy": all_acc,
    "F1 Score": all_f1
})

summary_df
```

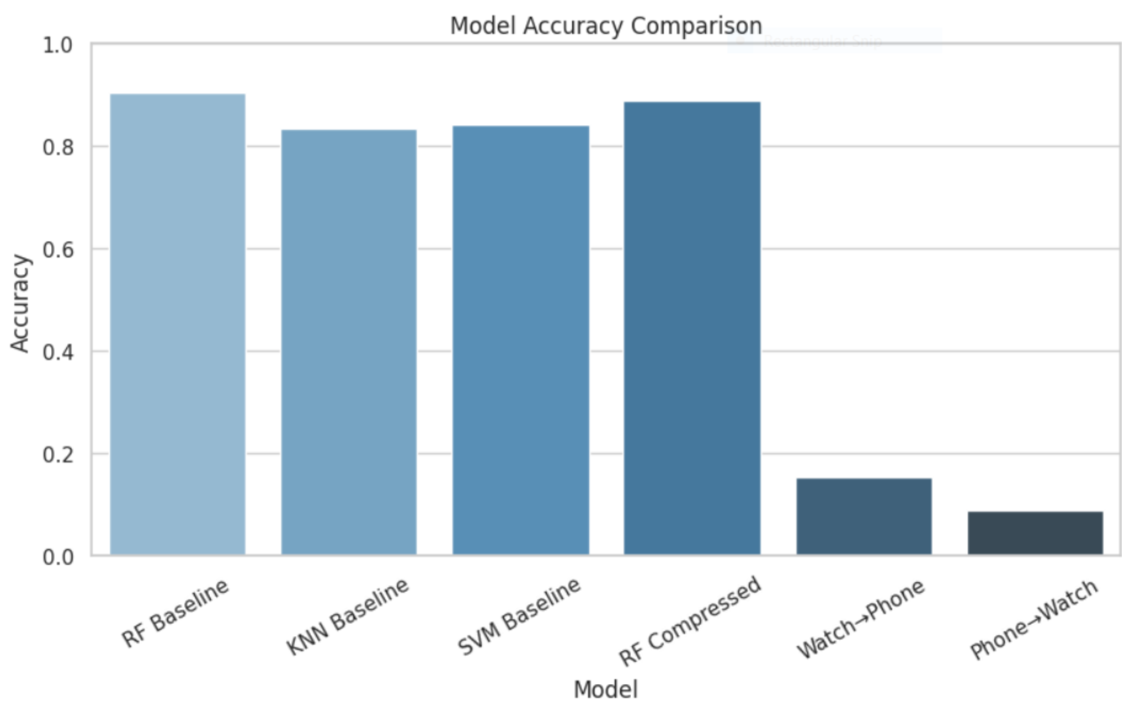
	Model	Accuracy	F1 Score
0	RF Baseline	0.903846	0.903174
1	KNN Baseline	0.834615	0.832643
2	SVM Baseline	0.842308	0.839313
3	RF Compressed	0.888462	0.888458
4	Watch→Phone	0.155521	0.098532
5	Phone→Watch	0.088666	0.065565

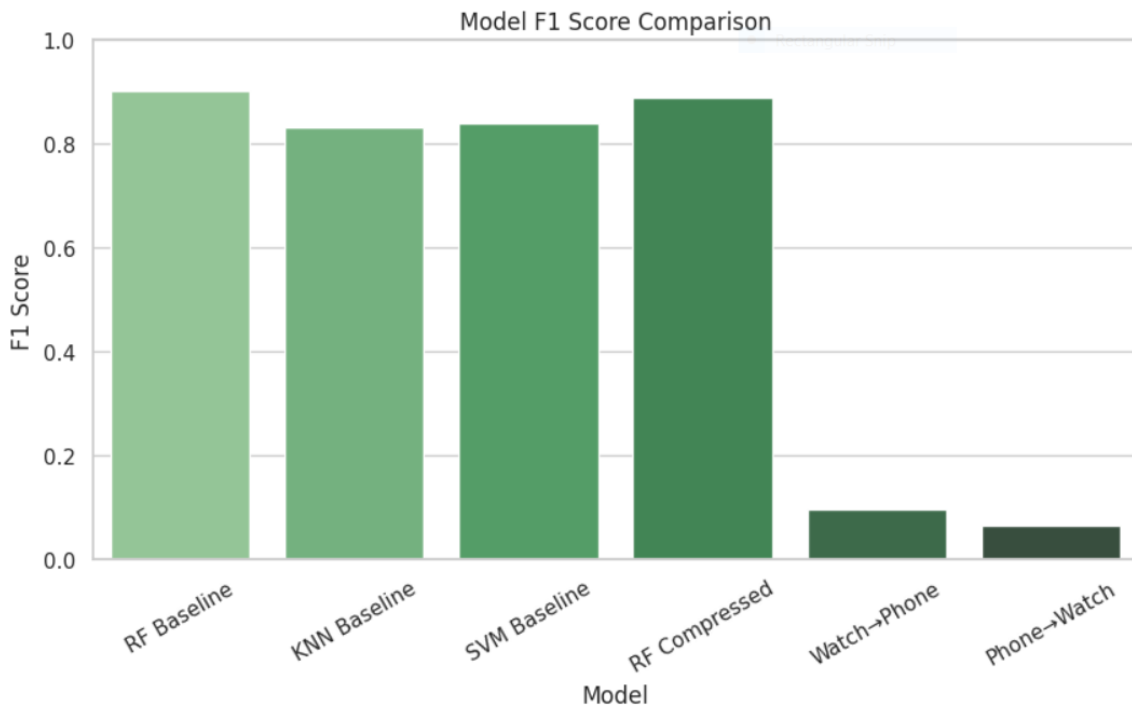
```python
import matplotlib.pyplot as plt
import seaborn as sns

# Set style
sns.set(style="whitegrid")
x_labels = summary_df["Model"]

# Accuracy Bar Plot
plt.figure(figsize=(10,5))
sns.barplot(x=x_labels, y=summary_df["Accuracy"], palette="Blues_d")
plt.ylim(0,1)
plt.title("Model Accuracy Comparison")
plt.ylabel("Accuracy")
plt.xlabel("Model")
plt.xticks(rotation=30)
plt.show()

# F1 Score Bar Plot
plt.figure(figsize=(10,5))
sns.barplot(x=x_labels, y=summary_df["F1 Score"], palette="Greens_d")
plt.ylim(0,1)
plt.title("Model F1 Score Comparison")
plt.ylabel("F1 Score")
plt.xlabel("Model")
plt.xticks(rotation=30)
plt.show()
```




Model F1 Score Comparison
F1 Score
1.0
0.8
0.6
0.4
0.2
0.0
RF Baseline
KNN Baseline
SVM Baseline
RF Compressed
Watch→Phone
Phone→Watch
Model