```
%pip install fpdf
```

Show hidden output

```
import warnings
warnings.filterwarnings('ignore')

import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt
%matplotlib inline

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.svm import SVR
from sklearn.preprocessing import PolynomialFeatures, StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.metrics import mean_squared_error, r2_score

from fpdf import FPDF

sns.set(style='whitegrid', palette='muted', font_scale=1.1)
print("Libraries loaded")
```

```
Libraries loaded
```

```
# Load Dataset
# -----------------------------
df = pd.read_csv("/content/Student_Marks.csv")
print(df.head())

X = df[['number_courses', 'time_study']]
y = df['Marks']

n = len(y)
print("Number of data points: ", n)
```

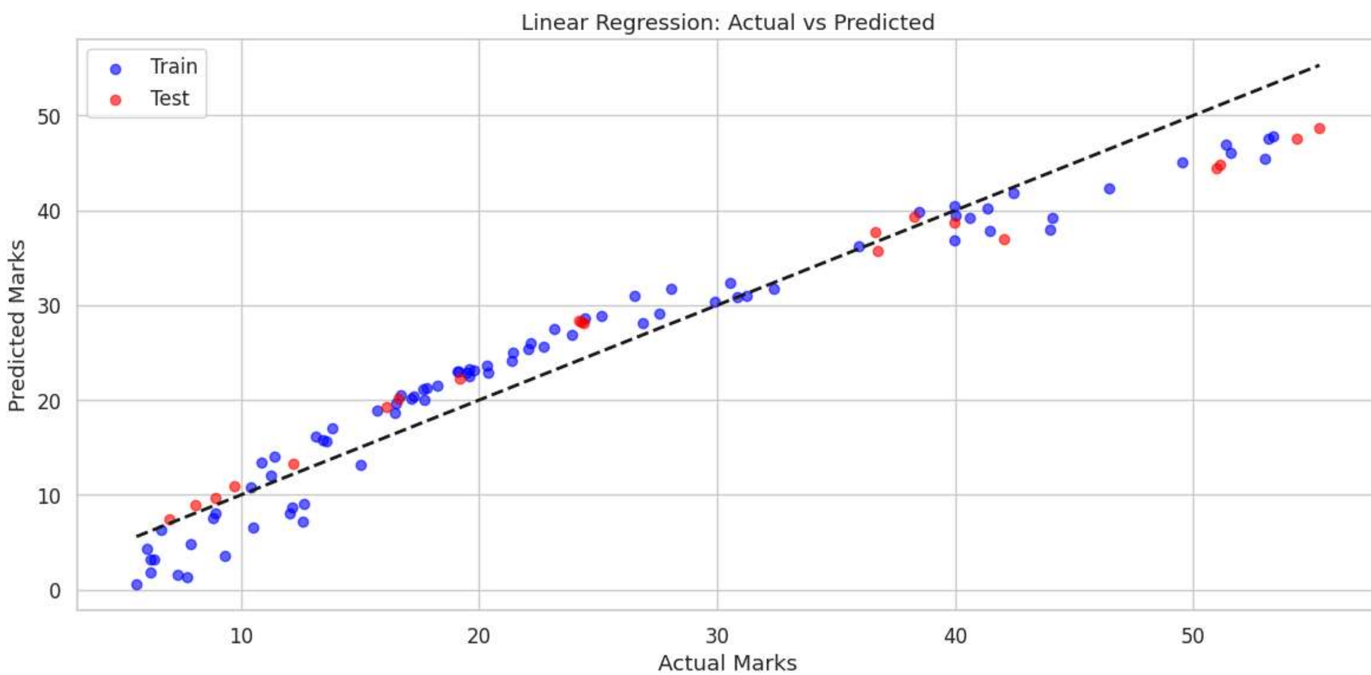
```
   number_courses  time_study   Marks
0               3       4.508  19.202
1               4       0.096   7.734
2               4       3.133  13.811
3               6       7.909  53.018
4               8       7.811  55.299
Number of data points:  100
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

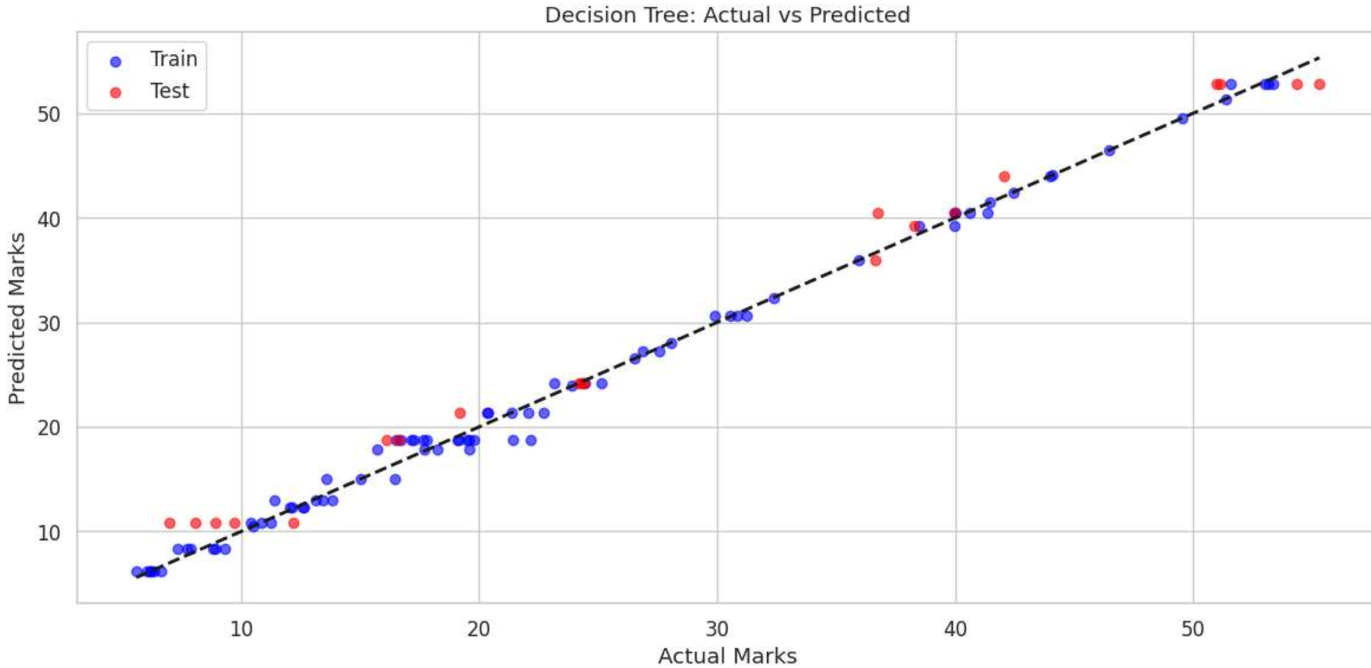
```
#  Helper Functions
# -----------------------------
def get_metrics(y_true, y_pred, n, k):
    r2 = r2_score(y_true, y_pred)
    adj_r2 = 1 - (1 - r2) * (n - 1) / (n - k - 1)
    rss = np.sum((y_true - y_pred) ** 2)
    mse = mean_squared_error(y_true, y_pred)
    rmse = np.sqrt(mse)
    aic = n * np.log(rss / n) + 2 * k
    return round(r2, 4), round(adj_r2, 4), round(rmse, 4), round(rss, 4), round(aic, 4)


def plot_actual_vs_pred(y_train, y_pred_train, y_test, y_pred_test, title, filename):
    plt.figure(figsize=(12, 6))
    plt.scatter(y_train, y_pred_train, color='blue', alpha=0.6, label='Train')
    plt.scatter(y_test, y_pred_test, color='red', alpha=0.6, label='Test')
    plt.plot([y.min(), y.max()], [y.min(), y.max()], 'k--', lw=2)
    plt.xlabel("Actual Marks")
    plt.ylabel("Predicted Marks")
    plt.title(title)
    plt.legend()
    plt.tight_layout()
```

What can I help you build?

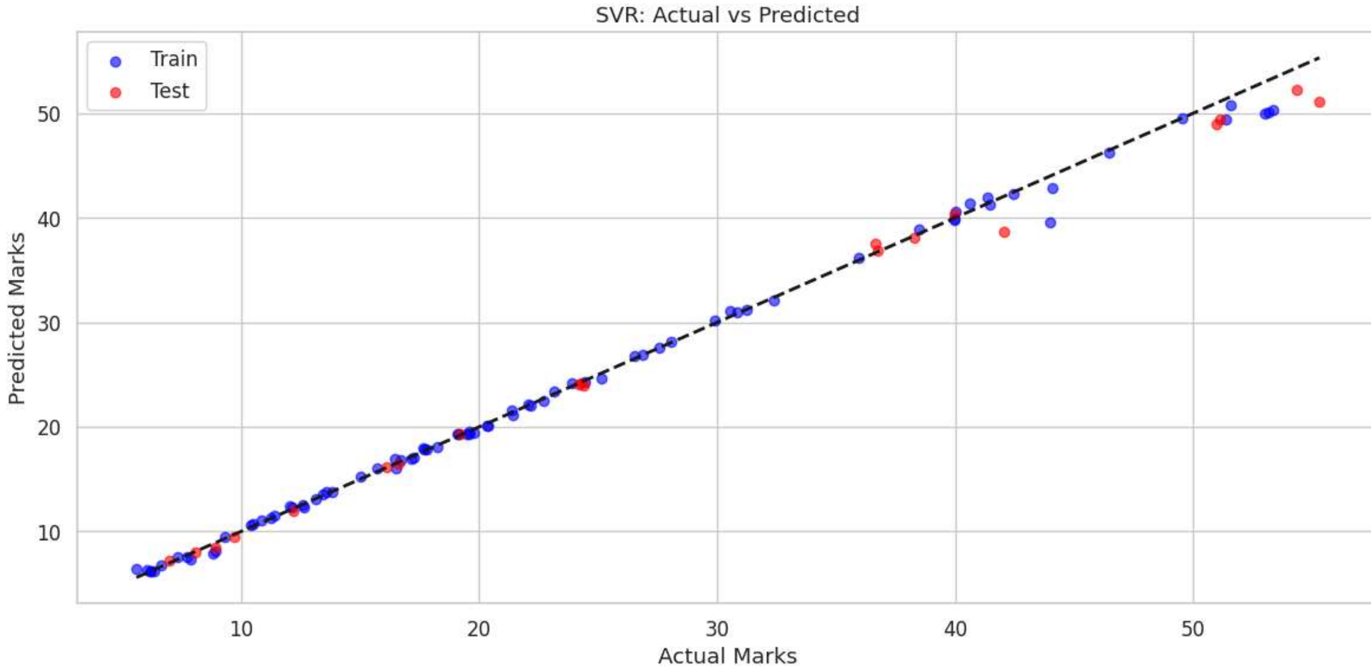
```
    plt.savefig(filename)
    plt.show()


# 1 Linear Regression
# -----------------------------
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)

y_pred_train_lr = lr_model.predict(X_train)
y_pred_test_lr = lr_model.predict(X_test)

lr_train_r2 = r2_score(y_train, y_pred_train_lr)
lr_test_r2, lr_adj_r2, lr_rmse, lr_rss, lr_aic = get_metrics(y_test, y_pred_test_lr, len(y_test), X_train.shape[1])

plot_actual_vs_pred(
    y_train, y_pred_train_lr,
    y_test, y_pred_test_lr,
    "Linear Regression: Actual vs Predicted",
    "lr_actual_vs_pred.png"
)
```




```
# 2 Decision Tree
# -----------------------------
dt_model = DecisionTreeRegressor(max_depth=5, random_state=42)
dt_model.fit(X_train, y_train)

y_pred_train_dt = dt_model.predict(X_train)
y_pred_test_dt = dt_model.predict(X_test)

dt_train_r2 = r2_score(y_train, y_pred_train_dt)
dt_test_r2, dt_adj_r2, dt_rmse, dt_rss, dt_aic = get_metrics(y_test, y_pred_test_dt, len(y_test), X_train.shape[1])

plot_actual_vs_pred(
    y_train, y_pred_train_dt,
    y_test, y_pred_test_dt,
    "Decision Tree: Actual vs Predicted",
    "dt_actual_vs_pred.png"
)
```




```
# 3 SVR
# ----------------------------
svr_pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('regressor', SVR(kernel='rbf', C=10, epsilon=0.2))
])
svr_pipeline.fit(X_train, y_train)

y_pred_train_svr = svr_pipeline.predict(X_train)
y_pred_test_svr = svr_pipeline.predict(X_test)

svr_train_r2 = r2_score(y_train, y_pred_train_svr)
svr_test_r2, svr_adj_r2, svr_rmse, svr_rss, svr_aic = get_metrics(y_test, y_pred_test_svr, len(y_test), X_train.shape[1])

plot_actual_vs_pred(
    y_train, y_pred_train_svr,
    y_test, y_pred_test_svr,
    "SVR: Actual vs Predicted",
    "svr_actual_vs_pred.png"
)
```




```
# 4 Polynomial Regression (Degrees 1-7)
# -----------------------------
degrees = list(range(1, 8))
poly_train_r2, poly_test_r2, poly_adj_r2, poly_rmse, poly_rss, poly_aic, poly_coeff, poly_icpt = [], [], [], [], [], [], [], []

for degree in degrees:
    poly_pipeline = Pipeline([
        ('poly_features', PolynomialFeatures(degree=degree, include_bias=False)),
        ('scaler', StandardScaler()),
        ('regressor', LinearRegression())
    ])
    poly_pipeline.fit(X_train, y_train)

    y_pred_train_poly = poly_pipeline.predict(X_train)
    y_pred_test_poly = poly_pipeline.predict(X_test)

    train_r2 = r2_score(y_train, y_pred_train_poly)
    test_r2, adj_r2, rmse, rss, aic = get_metrics(y_test, y_pred_test_poly, len(y_test), poly_pipeline.named_steps['poly_features'].n_output

    poly_train_r2.append(train_r2)
    poly_test_r2.append(test_r2)
    poly_adj_r2.append(adj_r2)
    poly_rmse.append(rmse)
    poly_rss.append(rss)
    poly_aic.append(aic)
    poly_coeff.append(poly_pipeline.named_steps['regressor'].coef_)
    poly_icpt.append(poly_pipeline.named_steps['regressor'].intercept_)


#  Polynomial Summary Table
# -----------------------------
pd.set_option('display.max_colwidth', None)

poly_df = pd.DataFrame({
    'Degree': degrees,
    'Train R2': poly_train_r2,
    'Test R2': poly_test_r2,
    'Adjusted R2': poly_adj_r2,
    'RMSE': poly_rmse,
    'RSS': poly_rss,
    'AIC': poly_aic,
    'Coefficients': poly_coeff,
    'Intercept': poly_icpt
})
```

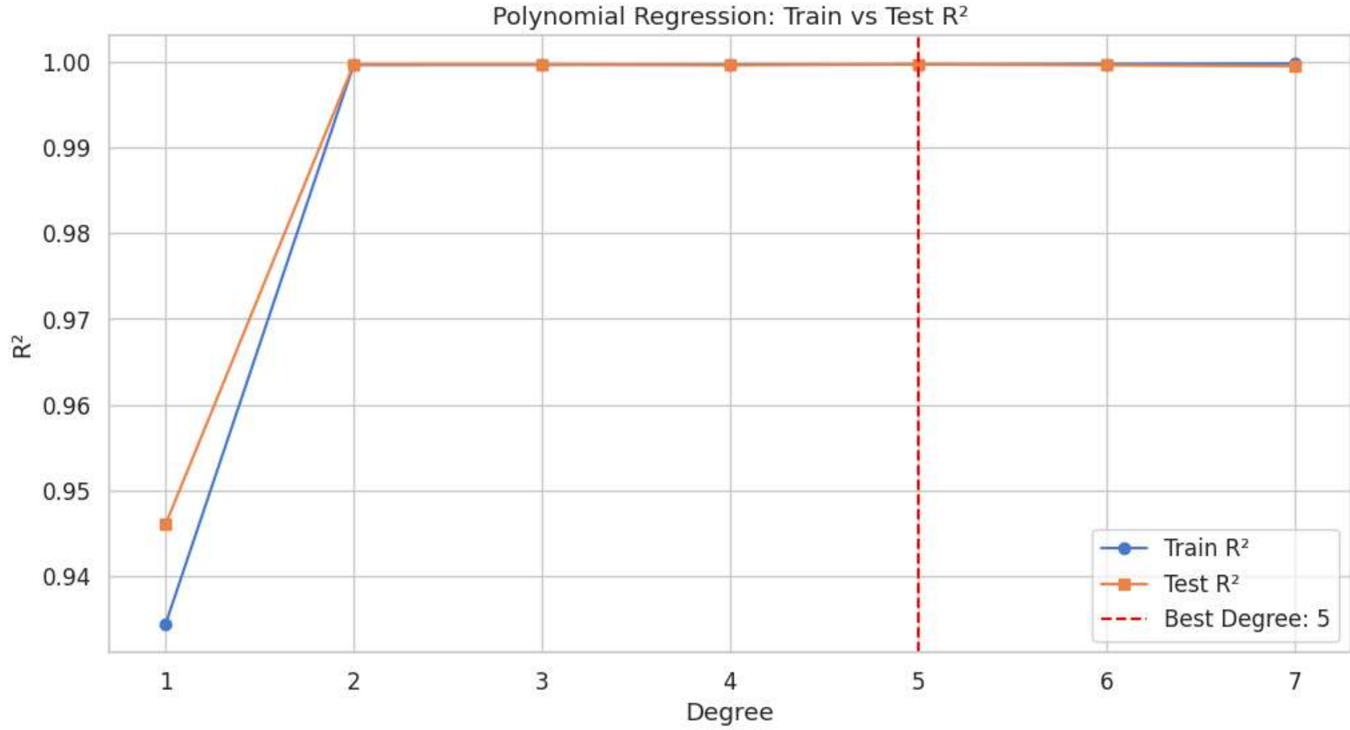
```
display(poly_df)
```

	Degree	Train R2	Test R2	Adjusted R2	RMSE	RSS	AIC	Coefficients	Intercept
0	1	0.934352	0.9460	0.9396	3.7684	284.0145	57.0659	[3.34457733290494, 12.182811923744508]	23.31945
1	2	0.999630	0.9997	0.9996	0.2853	1.6275	-40.1734	[2.988959090645139, -0.1933173434949631, -0.012540751708038256, 0.16146593559612, 12.762586577822484]	23.31945
2	3	0.999642	0.9997	0.9994	0.2859	1.6350	-32.0824	[3.5037040610775403, -0.20246598507541425, -1.354679883082043, 0.757584091237912, 12.147317447765724, 0.8116107888273821, -0.25660108799613623, -0.2763632160103677, 0.5596884105808317]	23.31945
3	4	0.999674	0.9996	0.9986	0.3121	1.9478	-18.5807	[-1.0807020113125327, 0.6117711189605517, 18.282800706648622, -7.805730695102624, 17.06334729968074, -26.91573299752396, 14.70829678728111, -1.859110142178094, -5.765974243316283, 12.847518366492832, -8.552380123871604, 2.044527330448502, -0.7477655691956386, 3.1675627902266674]	23.31945
4	5	0.999689	0.9997	1.0062	0.2930	1.7167	-9.1063	[19.81669687394436, 3.2730252302297793, -69.88373815355878, -21.20600692558814, 19.06188471834995, 120.77834943691927, 29.84813661525311, 15.58999622719085, -21.047800104884697, -102.94236771394336, -3.8335335907384342, -33.757557798858976, 10.12596411335815, 14.46631769551382, 35.459297424266346, -9.154185222776196, 18.670082591293305, -2.271776517401457, -3.895644368002976, -2.9667502234931202]	23.31945
								[-62.795537057120214, -44.44952641337596,	

Next steps: Generate code with poly_df | View recommended plots | New interactive sheet

```
# Best Polynomial Degree
best_poly_row = poly_df.sort_values(by=['Adjusted R2', 'RMSE', 'AIC'], ascending=[False, True, True]).iloc[0]
best_degree = int(best_poly_row['Degree'])
print(f" Best Polynomial Degree: {best_degree}")

plt.figure(figsize=(12, 6))
plt.plot(degrees, poly_train_r2, marker='o', label='Train R²')
plt.plot(degrees, poly_test_r2, marker='s', label='Test R²')
plt.axvline(x=best_degree, color='red', linestyle='--', label=f'Best Degree: {best_degree}')
plt.title("Polynomial Regression: Train vs Test R²")
plt.xlabel("Degree")
plt.ylabel("R²")
plt.legend()
plt.grid(True)
plt.savefig("poly_train_test_r2.png")
plt.show()
```

```
Best Polynomial Degree: 5
```



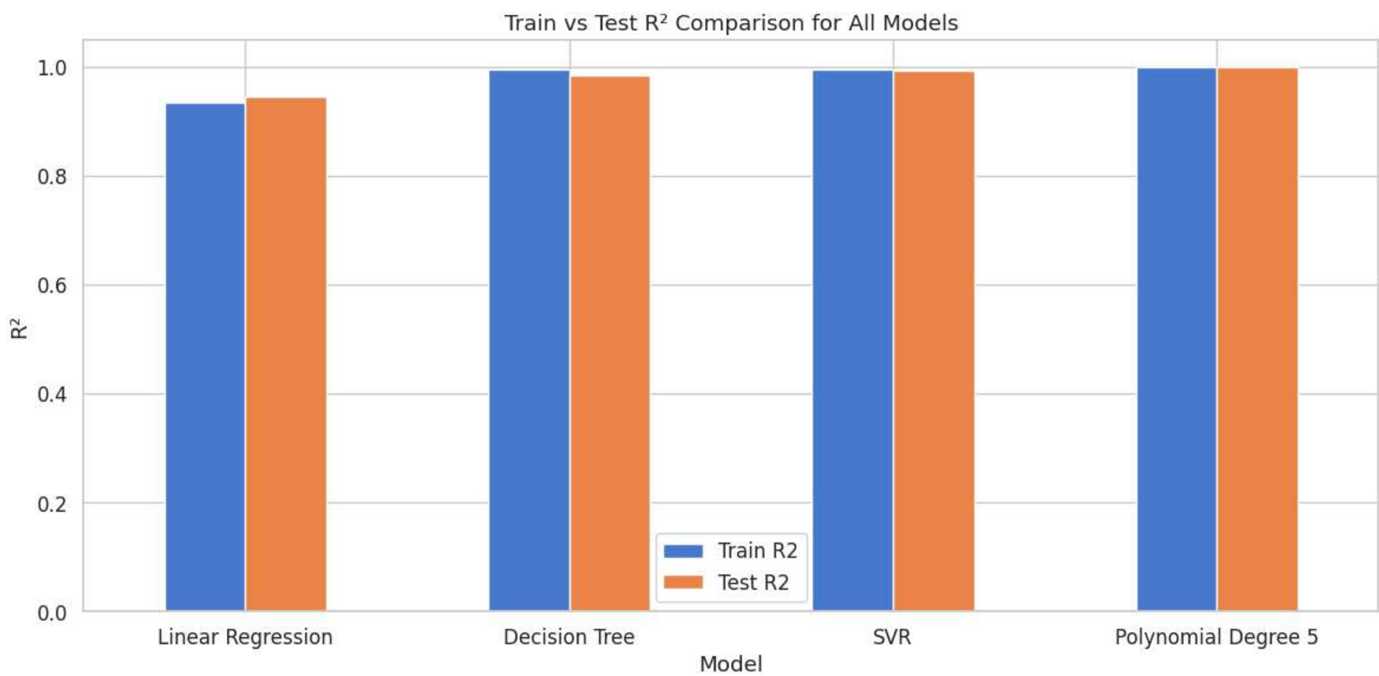

```
# Combined Model Summary
# ----------------------------
summary_df = pd.DataFrame({
    'Model': [
        'Linear Regression',
        'Decision Tree',
        'SVR',
        f'Polynomial Degree {best_degree}
    ],
    'Train R2': [
        lr_train_r2,
        dt_train_r2,
        svr_train_r2,
        poly_train_r2[best_degree - 1]
    ],
    'Test R2': [
        lr_test_r2,
        dt_test_r2,
        svr_test_r2,
        poly_test_r2[best_degree - 1]
    ],
    'Adjusted R2': [
        lr_adj_r2,
        dt_adj_r2,
        svr_adj_r2,
        poly_adj_r2[best_degree - 1]
    ],
    'RMSE': [
        lr_rmse,
        dt_rmse,
        svr_rmse,
        poly_rmse[best_degree - 1]
    ],
    'RSS': [
        lr_rss,
        dt_rss,
        svr_rss,
        poly_rss[best_degree - 1]
    ],
    'AIC': [
        lr_aic,
        dt_aic,
        svr_aic,
        poly_aic[best_degree - 1]
    ]
```

```
})

display(summary_df)
```

	Model	Train R2	Test R2	Adjusted R2	RMSE	RSS	AIC
0	Linear Regression	0.934352	0.[illegible]60	0.9396	3.7684	284.0145	57.0659
1	Decision Tree	0.995292	0.[illegible]49	0.9831	1.9920	79.3646	31.5664
2	SVR	0.995888	0.[illegible]20	0.9911	1.4481	41.9385	18.8095
3	Polynomial Degree 5	0.999689	0.[illegible]97	1.0062	0.2930	1.7167	-9.1063

Next steps: Generate code with summary_df View recommended plots New interactive sheet

```
summary_df.set_index('Model')[['Train R2', 'Test R2']].plot(kind='bar', figsize=(12, 6))
plt.title("Train vs Test R² Comparison for All Models")
plt.ylabel("R²")
plt.xticks(rotation=0)
plt.tight_layout()
plt.savefig("models_train_test_r2.png")
plt.show()

top_2_models = summary_df.sort_values(by=['Adjusted R2', 'RMSE'], ascending=[False, True]).head(2)
print("\n Top 2 Models to Use:")
display(top_2_models)
```




```
Top 2 Models to Use:
```