


```
1 import pandas as pd
2 import numpy as np
3 from sklearn.model_selection import train_test_split
4 from sklearn.linear_model import LogisticRegression
5 from sklearn.tree import DecisionTreeClassifier
6 from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
7 from sklearn.svm import SVC
8 from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score, roc_curve, confusion_matrix
9 from sklearn.inspection import permutation_importance
10 import matplotlib.pyplot as plt
11 from google.colab import files

1 uploaded = files.upload()
2 df = pd.read_csv("telco_encoded.csv")
3 df.head()
4
```

 Choose files

No file chosen

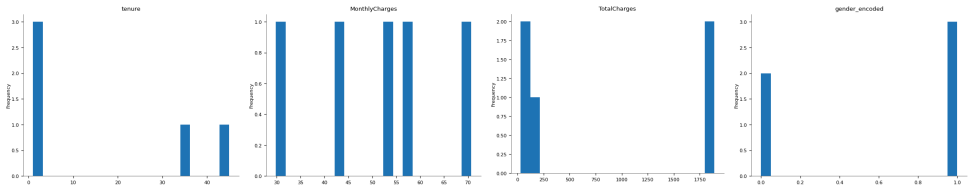
Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

Saving telco_encoded.csv to telco_encoded.csv

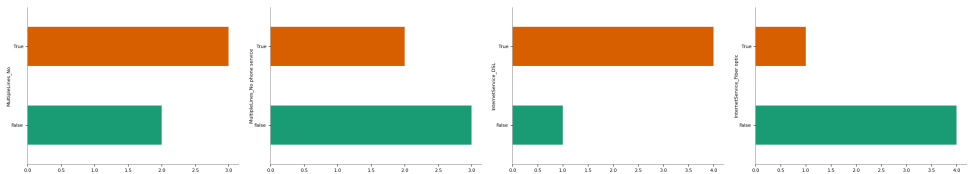
	SeniorCitizen	tenure	MonthlyCharges	TotalCharges	gender_encoded	Partner_encoded	Dependents_encoded	PhoneService_encoded	Paper...
0		0	1	29.85	29.85	0	1	0	0
1		0	34	56.95	1889.50	1	0	0	1
2		0	2	53.85	108.15	1	0	0	1
3		0	45	42.30	1840.75	1	0	0	0
4		0	2	70.70	151.65	0	0	0	1

5 rows × 41 columns

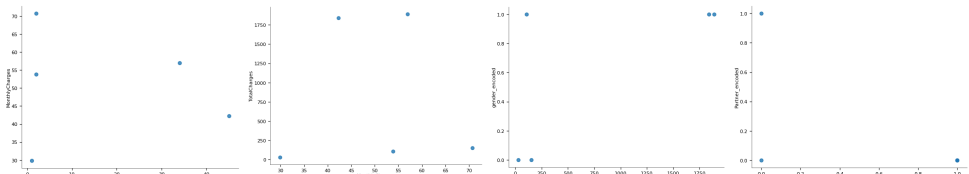
Distributions



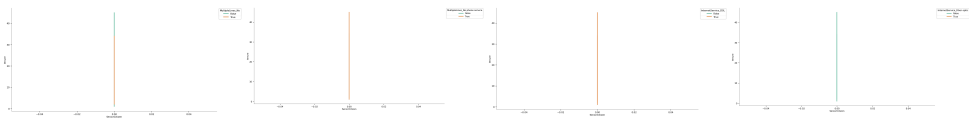
Categorical distributions



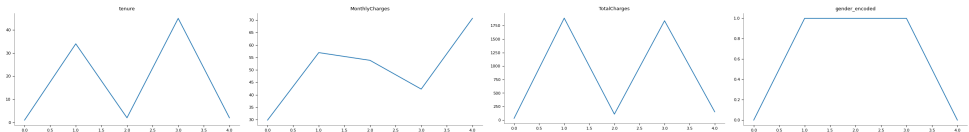
2-d distributions



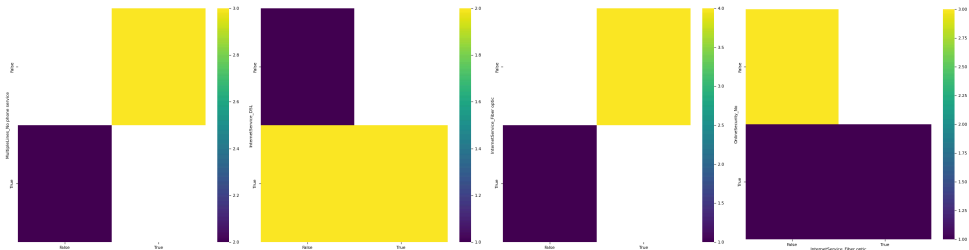
Time series



Values



2-d categorical distributions



Faceted distributions

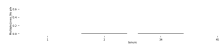
<string>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend`

<string>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend`

<string>:5: FutureWarning:



Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend`

```
<string>:5: FutureWarning:
```



Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend`



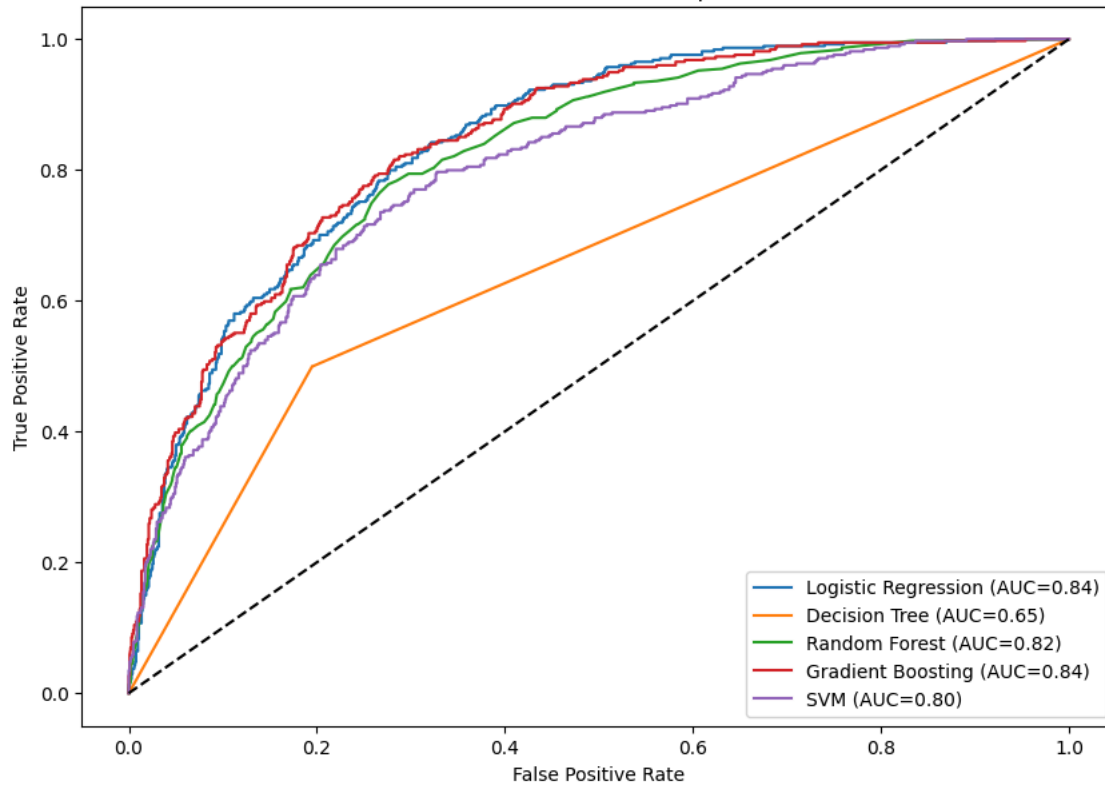
```
1 X = df.drop("Churn_encoded", axis=1)
2 y = df["Churn_encoded"]
3
4 X_train, X_test, y_train, y_test = train_test_split(
5     X, y, test_size=0.2, random_state=42, stratify=y
6 )
7

1 models = {
2     "Logistic Regression": LogisticRegression(max_iter=10000), # Increased max_iter
3     "Decision Tree": DecisionTreeClassifier(),
4     "Random Forest": RandomForestClassifier(),
5     "Gradient Boosting": GradientBoostingClassifier(),
6     "SVM": SVC(probability=True)
7 }

1 results = []
2 plt.figure(figsize=(10, 7))
3
4 for name, model in models.items():
5     model.fit(X_train, y_train)
6     y_pred = model.predict(X_test)
7     y_prob = model.predict_proba(X_test)[:, 1]
8
9     acc = accuracy_score(y_test, y_pred)
10    prec = precision_score(y_test, y_pred, zero_division=0)
11    rec = recall_score(y_test, y_pred, zero_division=0)
12    f1 = f1_score(y_test, y_pred, zero_division=0)
13    auc = roc_auc_score(y_test, y_prob)
14
15    results.append([name, acc, prec, rec, f1, auc])
16
17    fpr, tpr, _ = roc_curve(y_test, y_prob)
18    plt.plot(fpr, tpr, label=f"{name} (AUC={auc:.2f})")
19
20 plt.plot([0, 1], [0, 1], "k--")
21 plt.xlabel("False Positive Rate")
22 plt.ylabel("True Positive Rate")
23 plt.title("ROC Curves - Model Comparison")
24 plt.legend()
25 plt.show()
26
```



ROC Curves - Model Comparison



```

1 df_results = pd.DataFrame(results, columns=["Model", "Accuracy", "Precision", "Recall", "F1-Score", "AUC"])
2 df_results = df_results.set_index("Model")
3 display(df_results)
4
5 best_model_name = df_results["AUC"].idxmax()
6 print("Best model by AUC:", best_model_name)
7

```



	Accuracy	Precision	Recall	F1-Score	AUC
Model					
Logistic Regression	0.803407	0.652997	0.553476	0.599132	0.842515
Decision Tree	0.726757	0.485255	0.483957	0.484605	0.648976
Random Forest	0.785664	0.620000	0.497326	0.551929	0.821724
Gradient Boosting	0.804826	0.670103	0.521390	0.586466	0.843248
SVM	0.734564	0.000000	0.000000	0.000000	0.798884

Best model by AUC: Gradient Boosting

```

1 best_model = models[best_model_name]
2 y_pred_best = best_model.predict(X_test)
3 cm = confusion_matrix(y_test, y_pred_best)
4 print("Confusion Matrix (rows=true, cols=pred):")
5 print(cm)
6

```



```

Confusion Matrix (rows=true, cols=pred):
[[939  96]
 [179 195]]

```

```

1 rf = models["Random Forest"]
2 importances = rf.feature_importances_
3 feat_imp = pd.Series(importances, index=X.columns).sort_values(ascending=False)
4 print("Top 15 features by Random Forest importance:")
5 display(feat_imp.head(15))
6
7 perm = permutation_importance(rf, X_test, y_test, n_repeats=20, random_state=42)
8 perm_imp = pd.Series(perm.importances_mean, index=X.columns).sort_values(ascending=False)

```