randomforest

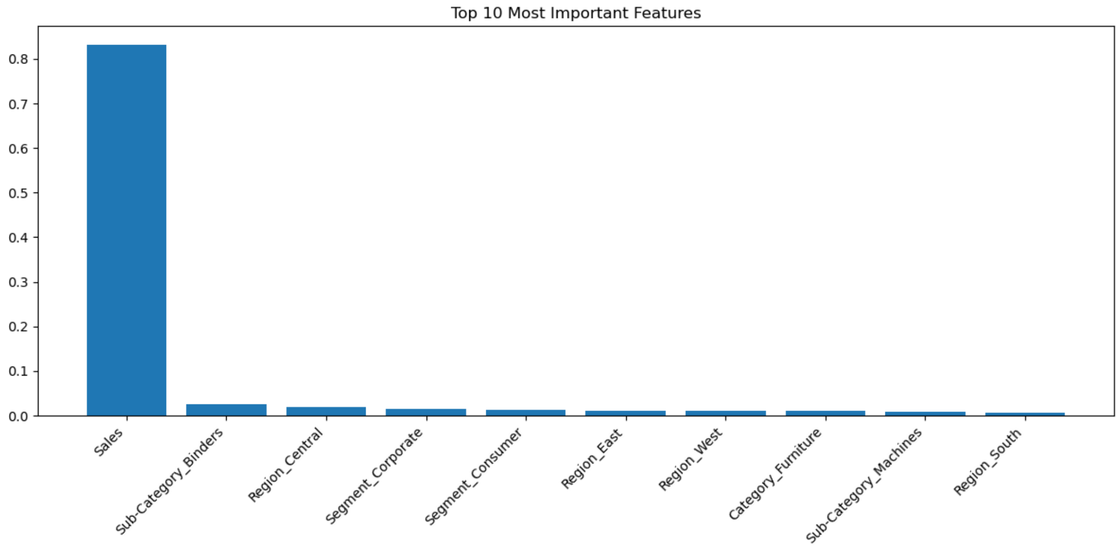
November 20, 2024

[1]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestRegressor
from datetime import datetime, timedelta
import seaborn as sns
```

0.0.1 Here we will provide a function for our Random Forest Regressor

[2]:
```
def train_eval_rf():
    try:
        # Read data
        df = pd.read_csv('Superstore.csv', encoding='ISO-8859-1')

        # Create features
        features = ['Sales', 'Category', 'Sub-Category', 'Region', 'Segment']

        # Convert categorical variables to dummy variables
        df_encoded = pd.get_dummies(df[features], columns=['Category',␣
↪'Sub-Category', 'Region', 'Segment'])

        # Prepare X and y
        X = df_encoded
        y = df['Profit']

        # Split data
        X_train, X_test, y_train, y_test = train_test_split(
            X, y, test_size=0.2, random_state=42
        )

        # Create and train model
        rf_model = RandomForestRegressor(
            n_estimators=100,
            max_depth=None,
            min_samples_split=2,
            min_samples_leaf=1,
            random_state=42
        )

        rf_model.fit(X_train, y_train)

        # Make predictions
        y_pred_train = rf_model.predict(X_train)
        y_pred_test = rf_model.predict(X_test)

        # Calculate scores
        train_score = r2_score(y_train, y_pred_train)
        test_score = r2_score(y_test, y_pred_test)

        # Get feature importance
        feature_importance = pd.DataFrame({
            'feature': X_train.columns,
            'importance': rf_model.feature_importances_
        }).sort_values('importance', ascending=False)

        # Create feature importance plot
        plt.figure(figsize=(12, 6))
        plt.bar(feature_importance['feature'][:10], feature_importance['importance'][:10])
        plt.xticks(rotation=45, ha='right')
        plt.title('Top 10 Most Important Features')
        plt.tight_layout()

        # Return the values
        return train_score, test_score, feature_importance

    except Exception as e:
        print(f"An error occurred: {str(e)}")
        return None, None, None

# Run the analysis
train_score, test_score, feature_importance = train_eval_rf()

# Check if we got valid results
if train_score is not None:
    print(f"\nRandom Forest Results:")
    print(f"Training R² score: {train_score:.4f}")
    print(f"Testing R² score: {test_score:.4f}")
    print("\nTop 5 Most Important Features:")
    print(feature_importance.head())
    plt.show()
else:
    print("Failed to run the analysis. Please check your data and try again.")
```

```
Random Forest Results:
Training R² score: 0.9120
Testing R² score: -0.6703

Top 5 Most Important Features:
                 feature  importance
0                  Sales    0.831803
7    Sub-Category_Binders    0.026070
21        Region_Central    0.018102
26     Segment_Corporate    0.014203
25      Segment_Consumer    0.012612
```



0.0.2 We are going to tweak our model a bit and see if we can get a better mix of training and test results.

[ ]:

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
from sklearn.metrics import roc_curve, auc
```

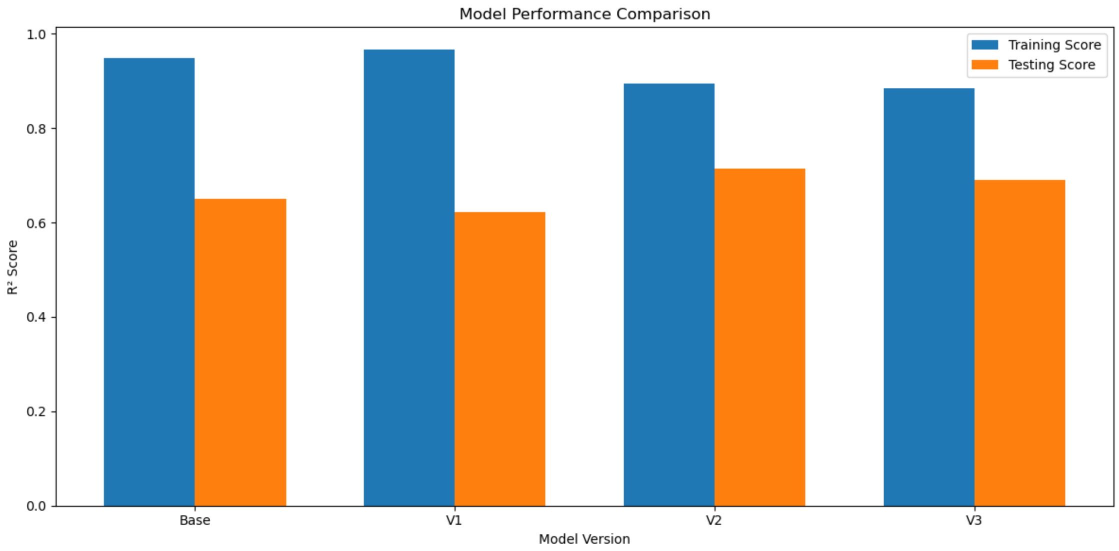
[4]:

```
def train_eval_rf_models():
    # Read and prepare data
    df = pd.read_csv('Superstore.csv', encoding='ISO-8859-1')

    # Feature engineering
```

```
df['Profit_Margin'] = df['Profit'] / df['Sales']
df['Discount_Amount'] = df['Sales'] * df['Discount']

features = [
    'Sales', 'Discount', 'Quantity',
    'Profit_Margin', 'Discount_Amount',
    'Category', 'Sub-Category', 'Region',
    'Segment', 'Ship Mode'
]

# Prepare data
df_encoded = pd.get_dummies(df[features],
                          columns=['Category', 'Sub-Category',
                                 'Region', 'Segment', 'Ship Mode'])
X = df_encoded
y = df['Profit']

# Split data
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# Define models
rf_model_base = RandomForestRegressor(
    max_depth=None,
    max_features="sqrt",
    max_leaf_nodes=None,
    min_samples_leaf=1,
    min_samples_split=2,
    min_weight_fraction_leaf=0.0,
    n_estimators=8,
    n_jobs=1,
    oob_score=False,
    random_state=None,
    verbose=0,
    warm_start=False
)

rf_model_V1 = RandomForestRegressor(
    n_estimators=25,
    max_features="sqrt",
    bootstrap=True,
    oob_score=True,
    random_state=42
)

rf_model_V2 = RandomForestRegressor(
    n_estimators=15,
    max_features="sqrt",
    min_samples_leaf=2,
    min_samples_split=3,
    random_state=42
)

rf_model_V3 = RandomForestRegressor(
    n_estimators=20,
    max_features="sqrt",
    max_leaf_nodes=None,
    min_samples_leaf=2,
    bootstrap=True,
    random_state=42
)

# Dictionary to store results
results = {}

# Train and evaluate each model
for name, model in [
    ('Base', rf_model_base),
    ('V1', rf_model_V1),
    ('V2', rf_model_V2),
    ('V3', rf_model_V3)
]:
    # Train model
    model.fit(X_train, y_train)

    # Make predictions
    y_pred_train = model.predict(X_train)
    y_pred_test = model.predict(X_test)

    # Calculate scores
    train_score = r2_score(y_train, y_pred_train)
    test_score = r2_score(y_test, y_pred_test)



    # Store feature importance
    feature_importance = pd.DataFrame({
        'feature': X_train.columns,
        'importance': model.feature_importances_
    }).sort_values('importance', ascending=False)

    results[name] = {
        'train_score': train_score,
            'test_score': test_score,
            'feature_importance': feature_importance
        }

    return results

# Run all models and get results
results = train_eval_rf_models()

# Print results
for model_name, model_results in results.items():
    print(f"\nModel {model_name} Results:")
    print(f"Training R² score: {model_results['train_score']:.4f}")
    print(f"Testing R² score: {model_results['test_score']:.4f}")
    print("\nTop 5 Most Important Features:")
    print(model_results['feature_importance'].head())
    print("-" * 50)

# Visualize comparison
plt.figure(figsize=(12, 6))
x = np.arange(len(results))
width = 0.35

plt.bar(x - width/2, [r['train_score'] for r in results.values()], width,␣
  ↪label='Training Score')
plt.bar(x + width/2, [r['test_score'] for r in results.values()], width,␣
  ↪label='Testing Score')

plt.xlabel('Model Version')
plt.ylabel('R² Score')
plt.title('Model Performance Comparison')
plt.xticks(x, results.keys())
plt.legend()
plt.tight_layout()
plt.show()
```

```
Model Base Results:
Training R² score: 0.9479
Testing R² score: 0.6495

Top 5 Most Important Features:
                 feature  importance
0                  Sales    0.407304
4        Discount_Amount    0.153583
3          Profit_Margin    0.114142
14  Sub-Category_Copiers    0.072162
```

```
2               Quantity    0.066964
--------------------------------------------------

Model V1 Results:
Training R² score: 0.9655
Testing R² score: 0.6230

Top 5 Most Important Features:
                 feature  importance
0                  Sales    0.502322
4        Discount_Amount    0.121559
3          Profit_Margin    0.115381
14  Sub-Category_Copiers    0.047699
2               Quantity    0.040134
--------------------------------------------------

Model V2 Results:
Training R² score: 0.8949
Testing R² score: 0.7151

Top 5 Most Important Features:
                 feature  importance
0                  Sales    0.514651
4        Discount_Amount    0.168590
3          Profit_Margin    0.088355
14  Sub-Category_Copiers    0.066833
2               Quantity    0.031925
--------------------------------------------------

Model V3 Results:
Training R² score: 0.8844
Testing R² score: 0.6893

Top 5 Most Important Features:
                 feature  importance
0                  Sales    0.487046
4        Discount_Amount    0.152881
3          Profit_Margin    0.101438
14  Sub-Category_Copiers    0.069265
1               Discount    0.033524
--------------------------------------------------
```



```
[5]: def train_eval_rf_stable():
         # Set random seed for entire process
         np.random.seed(42)

         # Read and prepare data
         df = pd.read_csv('Superstore.csv', encoding='ISO-8859-1')

         # Feature engineering
         df['Profit_Margin'] = df['Profit'] / df['Sales']
         df['Discount_Amount'] = df['Sales'] * df['Discount']

         features = [
             'Sales', 'Discount', 'Quantity',
             'Profit_Margin', 'Discount_Amount',
             'Category', 'Sub-Category', 'Region',
             'Segment', 'Ship Mode'
         ]

         # Prepare data
         df_encoded = pd.get_dummies(df[features],
                                     columns=['Category', 'Sub-Category',
                                             'Region', 'Segment', 'Ship Mode'])
         X = df_encoded
         y = df['Profit']

         # Split data with fixed random state
         X_train, X_test, y_train, y_test = train_test_split(
             X, y, test_size=0.2, random_state=42
```

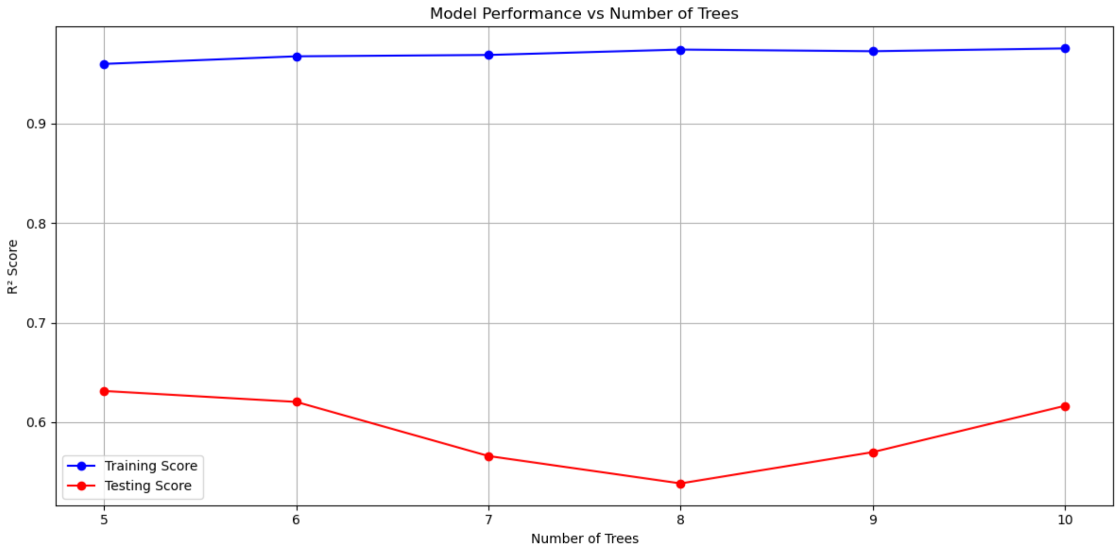
```
    )

    # Define model with fixed random state
    rf_model = RandomForestRegressor(
        max_depth=None,
        max_features="sqrt",
        max_leaf_nodes=None,
        min_samples_leaf=1,
        min_samples_split=2,
        min_weight_fraction_leaf=0.0,
        n_estimators=10,
        n_jobs=1,
        random_state=42,  # Fixed random state
        bootstrap=True    # Explicitly set bootstrap
    )

    # Train model
    rf_model.fit(X_train, y_train)

    # Make predictions
    y_pred_train = rf_model.predict(X_train)
    y_pred_test = rf_model.predict(X_test)

    # Calculate scores
    train_score = r2_score(y_train, y_pred_train)
    test_score = r2_score(y_test, y_pred_test)

    # Get feature importance
    feature_importance = pd.DataFrame({
        'feature': X_train.columns,
        'importance': rf_model.feature_importances_
    }).sort_values('importance', ascending=False)

    return train_score, test_score, feature_importance

# Run the model multiple times to verify stability
print("Running model multiple times to verify stability:\n")

for i in range(3):
    train_score, test_score, feature_importance = train_eval_rf_stable()
    print(f"Run {i+1}:")
    print(f"Training R² score: {train_score:.4f}")
    print(f"Testing R² score: {test_score:.4f}")
    print("\nTop 5 Most Important Features:")
    print(feature_importance.head())
    print("-" * 50 + "\n")
```

```
Running model multiple times to verify stability:
```

```
Run 1:
Training R² score: 0.9756
Testing R² score: 0.6164

Top 5 Most Important Features:
                 feature  importance
0                  Sales    0.498980
4        Discount_Amount    0.128803
3          Profit_Margin    0.100625
14  Sub-Category_Copiers    0.051662
2               Quantity    0.038025
--------------------------------------------------

Run 2:
Training R² score: 0.9756
Testing R² score: 0.6164

Top 5 Most Important Features:
                 feature  importance
0                  Sales    0.498980
4        Discount_Amount    0.128803
3          Profit_Margin    0.100625
14  Sub-Category_Copiers    0.051662
2               Quantity    0.038025
--------------------------------------------------

Run 3:
Training R² score: 0.9756
Testing R² score: 0.6164

Top 5 Most Important Features:
                 feature  importance
0                  Sales    0.498980
4        Discount_Amount    0.128803
3          Profit_Margin    0.100625
14  Sub-Category_Copiers    0.051662
2               Quantity    0.038025
--------------------------------------------------
```

```
[6]: def test_n_estimators(n_range):
         # Set random seed
         np.random.seed(42)

         # Read and prepare data
         df = pd.read_csv('Superstore.csv', encoding='ISO-8859-1')
```

```python
# Feature engineering
df['Profit_Margin'] = df['Profit'] / df['Sales']
df['Discount_Amount'] = df['Sales'] * df['Discount']

features = [
    'Sales', 'Discount', 'Quantity',
    'Profit_Margin', 'Discount_Amount',
    'Category', 'Sub-Category', 'Region',
    'Segment', 'Ship Mode'
]

# Prepare data
df_encoded = pd.get_dummies(df[features],
                          columns=['Category', 'Sub-Category',
                                 'Region', 'Segment', 'Ship Mode'])
X = df_encoded
y = df['Profit']

# Split data
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

results = []

# Test different numbers of trees
for n in n_range:
    rf_model = RandomForestRegressor(
        n_estimators=n,
        max_depth=None,
        max_features="sqrt",
        max_leaf_nodes=None,
        min_samples_leaf=1,
        min_samples_split=2,
        random_state=42
    )

    rf_model.fit(X_train, y_train)

    train_score = r2_score(y_train, rf_model.predict(X_train))
    test_score = r2_score(y_test, rf_model.predict(X_test))

    results.append({
        'n_estimators': n,
        'train_score': train_score,
        'test_score': test_score
        })

    return pd.DataFrame(results)

# Test range of trees around 7
n_range = [5, 6, 7, 8, 9, 10]
results_df = test_n_estimators(n_range)

# Plot results
plt.figure(figsize=(12, 6))
plt.plot(results_df['n_estimators'], results_df['train_score'],
         'b-o', label='Training Score')
plt.plot(results_df['n_estimators'], results_df['test_score'],
         'r-o', label='Testing Score')
plt.xlabel('Number of Trees')
plt.ylabel('R² Score')
plt.title('Model Performance vs Number of Trees')
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()

# Print detailed results
print("\nDetailed Results:")
print(results_df.round(4))

# Find best number of trees
best_test = results_df.loc[results_df['test_score'].idxmax()]
print(f"\nBest Performance:")
print(f"Number of Trees: {best_test['n_estimators']}")
print(f"Test Score: {best_test['test_score']:.4f}")
print(f"Train Score: {best_test['train_score']:.4f}")
```



```
Detailed Results:
   n_estimators  train_score  test_score
0             5       0.9600      0.6313
1             6       0.9677      0.6203
2             7       0.9691      0.5659
3             8       0.9745      0.5383
4             9       0.9728      0.5698
5            10       0.9756      0.6164

Best Performance:
Number of Trees: 5.0
Test Score: 0.6313
Train Score: 0.9600
```

```
[ ]:
```