预测代码.py

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import statsmodels.api as sm
from statsmodels.tsa.statespace.sarimax import SARIMAX
from datetime import datetime, timedelta
plt.rcParams['font.sans-serif'] = ['SimHei']
df = pd.read_excel("G:/mathematic/sales_summary.xlsx")
df['销售日期'] = pd.to_datetime(df['销售日期'])
category_mapping = {
    "花叶类": "1011010101",
    "花菜类": "1011010201",
    "水生根茎类": "1011010402",
    "茄类": "1011010501",
    "辣椒类": "1011010504",
    "食用菌": "1011010801"
}
fig, axes = plt.subplots(nrows=2, ncols=3, figsize=(15, 10))
plt.subplots_adjust(wspace=0.3, hspace=0.5)
for i, (category_name, category_code) in enumerate(category_mapping.items()):
    category_sales = df[df['分类名称'] == category_name][['销售日期', '总销量(千克)']].copy()
    category_sales.set_index('销售日期', inplace=True)
    daily_sales = category_sales.resample('D').sum()
    model = SARIMAX(daily_sales, order=(p, d, q))
```

```
    forecast = model_fit.forecast(steps=forecast_days)
    last_date = daily_sales.index[-1]
    date_range = [last_date + timedelta(days=i) for i in range(1, forecast_days + 1)]
    forecast_df = pd.DataFrame({'销售日期': date_range, '总销量(千克)': forecast})
    row = i // 3
    col = i % 3
    ax = axes[row, col]
    ax.plot(daily_sales.index, daily_sales['总销量(千克)'], label='历史销售量')
    ax.plot(forecast_df['销售日期'], forecast_df['总销量(千克)'], label='销售量预测', linestyle='--',
marker='o')
    ax.set_xlabel('销售日期')
    ax.set_ylabel('总销量(千克)')
    ax.set_title(f'未来一周总销量预测 - {category_name}')
    ax.legend()
combined_image_filename = "combined_sales_forecast.png"
plt.savefig(f"G:/mathematic/图片_1/{combined_image_filename}")
plt.show()
for category_name, category_code in category_mapping.items():
    print(f"分类名称: {category_name}")
    print(forecast_df)
    print("\n")
for category_name, category_code in category_mapping.items():
    forecast_df.to_csv(f"G:/mathematic/cvs/{category_name}_sales_forecast.csv", index=False)
```

随机森林代码.py

```
import pandas as pd
```

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import PolynomialFeatures
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score
from matplotlib.font_manager import FontProperties
category_mapping = {
    "1011010101": "花叶类",
    "1011010201": "花菜类",
    "1011010402": "水生根茎类",
    "1011010501": "茄类",
    "1011010504": "辣椒类",
    "1011010801": "食用菌"
}
for category_code, category_name in category_mapping.items():
    filtered_df = df[df["分类编码"] == int(category_code)]
    y_data = filtered_df["总销量(千克)"]
    poly_features = PolynomialFeatures(degree=degree)
    x_poly = poly_features.fit_transform(x_data)
    random_forest_model = RandomForestRegressor(n_estimators=n_estimators, random_state=0)
    random_forest_model.fit(x_poly, y_data)
    y_pred = random_forest_model.predict(x_poly)
    r_squared = r2_score(y_data, y_pred)
    feature_importance = random_forest_model.feature_importances_
    x_sorted = np.sort(x_data.iloc[:, 0])
    plt.scatter(x_data.iloc[:, 0], y_data, label="销售数据")
```

```
    plt.plot(x_sorted, y_pred[np.argsort(x_data.iloc[:, 0])], color="red", label="随机森林回归曲线
")

    plt.xlabel("销售平均单价(元/千克)", fontproperties=font)

    plt.ylabel("总销量(千克)", fontproperties=font)

    plt.title(f"{category_name} 随机森林回归分析", fontproperties=font)

    plt.legend(prop=font)

    plt.text(0.7, 0.7, f"{category_name} R 方: {r_squared:.2f}", transform=plt.gca().transAxes,
fontsize=10, fontproperties=font)

    image_filename = f"{category_name}_随机森林回归分析图.png"

    plt.savefig(f"G:/mathematic/图片/{image_filename}")

    plt.clf()
```

多元线性回归代码.py

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.preprocessing import PolynomialFeatures

from sklearn.linear_model import Ridge

from matplotlib.font_manager import FontProperties

category_mapping = {

    "1011010101": "花叶类",

    "1011010201": "花菜类",

    "1011010402": "水生根茎类",

    "1011010501": "茄类",

    "1011010504": "辣椒类",

    "1011010801": "食用菌"
```

```
}
for category_code, category_name in category_mapping.items():
    filtered_df = df[df["分类编码"] == int(category_code)]
    y_data = filtered_df["总销量(千克)"]
    poly_features = PolynomialFeatures(degree=degree)
    x_poly = poly_features.fit_transform(x_data)
    ridge_model = Ridge(alpha=alpha)
    ridge_model.fit(x_poly, y_data)
    y_pred = ridge_model.predict(x_poly)
    r_squared = ridge_model.score(x_poly, y_data)
    x_sorted = np.sort(x_data.iloc[:, 0])
    plt.scatter(x_data.iloc[:, 0], y_data, label="销售数据")
    plt.plot(x_sorted, y_pred[np.argsort(x_data.iloc[:, 0])], color="red", label="岭回归曲线")
    plt.xlabel("销售平均单价(元/千克)", fontproperties=font)
    plt.ylabel("总销量(千克)", fontproperties=font)
    plt.title(f"{category_name} 岭回归分析", fontproperties=font)
    plt.legend(prop=font)
    plt.text(0.1, 0.9, f"{category_name} R 方: {r_squared:.2f}", transform=plt.gca().transAxes,
fontsize=10, fontproperties=font)
    image_filename = f"{category_name}_岭回归分析图.png"
    plt.savefig(f'G:/mathematic/图片/{image_filename}')
    plt.clf()
```

非线性回归代码.py

```
import pandas as pd
import numpy as np
```

```
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from matplotlib.font_manager import FontProperties
category_mapping = {
    "1011010101": "花叶类",
    "1011010201": "花菜类",
    "1011010402": "水生根茎类",
    "1011010501": "茄类",
    "1011010504": "辣椒类",
    "1011010801": "食用菌"
}
for category_code, category_name in category_mapping.items():
    filtered_df = df[df["分类编码"] == int(category_code)]
    y_data = filtered_df["销量(千克)"]
    model = LinearRegression()
    model.fit(x_data, y_data)
    coefficients = model.coef_
    intercept = model.intercept_
    regression_equation = f"y = {intercept:.2f} + "
    for i, coef in enumerate(coefficients):
        regression_equation += f"{coef:.2f} * X{i+1} + "
    regression_equation = regression_equation[:-2]
    r_squared = model.score(x_data, y_data)
    plt.scatter(x_data.iloc[:, 0], y_data, label="销售数据")
    plt.plot(x_data.iloc[:, 0], model.predict(x_data), color="red", label="回归曲线")
    plt.xlabel("销售单价(元/千克)", fontproperties=font)
```

```
    plt.ylabel("销量(千克)", fontproperties=font)
    plt.title(f"{category_name} 多元线性回归分析", fontproperties=font)
    plt.legend(prop=font)
    text = f"{category_name} 回归方程:\n{regression_equation}\n\n{category_name} R 方:
{r_squared:.2f}"
    plt.text("", "", "", fontproperties=font)
    plt.text(0.5, 0.6, text, transform=plt.gca().transAxes, fontsize=10)
    image_filename = f"{category_name}_多元线性回归分析图.png"
    plt.savefig(f'G:/mathematic/图片/{image_filename}")
    plt.clf()
```

小波分析算法.py

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import pywt
df = pd.read_excel('/Users/mark/Desktop/merge.xlsx')
df['销售日期'] = pd.to_datetime(df['销售日期'])
df = df.set_index('销售日期')
groups = df.groupby('分类编码')
for name, group in groups:
    signal = group['销量(千克)']
    level = pywt.dwt_max_level(data_len=len(signal),
    filter_len=pywt.Wavelet('db2').dec_len)
    coeffs = pywt.wavedec(signal, 'db2', level=5)
    fig, ax = plt.subplots(len(coeffs), 1, figsize=(10, 20))
```

```
for i in range(len(coeffs)):
    ax[i].plot(coeffs[i])
    ax[i].set_title(f'Level {i+1}')
plt.tight_layout()
plt.show()
```

优化问题.py

```
import numpy as np
import pandas as pd
from scipy.optimize import linprog
df = pd.read_excel('G:/mathematic/sales_summary.xlsx')
df1 = pd.read_csv('G:/mathematic/cvs/花叶类_sales_forecast.csv')
product_codes = 1011010101
prices = df['销售平均单价(元/千克)'].tolist()
costs = df['批发价格(元/千克)'].tolist()
min_display = 2.5
sales_forecast = df1['总销量(千克)'].tolist()
c = np.array(prices) - np.array(costs) # 目标函数系数（最大化总收益）
A_eq = np.ones((1, len(int(product_codes)))) # 约束矩阵（总售卖单品数量）
b_eq = np.array([27]) # 约束条件（总售卖单品数量限制在 27-33 个之间）
bounds = [(0, None) for _ in product_codes] # 变量边界（补货量为非负数）
result = linprog(-c, A_eq=A_eq, b_eq=b_eq, bounds=bounds)
if result.success:
    print("Optimal Solution Found!")
    quantities = result.x
    for i, code in enumerate(product_codes):
```

```
        print(f"Product {code}: Quantity = {quantities[i]:.2f}")
    else:
        print("No Optimal Solution Found.")
    import numpy as np
    from scipy.optimize import linprog
    product_codes = ["1011010101", "1011010201", "1011010402", "1011010501", "1011010504",
"1011010801"]
    prices = [10, 8, 9, 12, 7, 6]
    costs = [4, 3, 4, 5, 2, 3]
    min_display = [2.5, 2.5, 2.5, 2.5, 2.5, 2.5]
    c = np.array(prices) - np.array(costs) # 目标函数系数（最大化总收益）
    A_eq = np.ones((1, len(product_codes))) # 约束矩阵（总售卖单品数量）
    b_eq = np.array([27]) # 约束条件（总售卖单品数量限制在 27-33 个之间）
    bounds = [(0, None) for _ in product_codes] # 变量边界（补货量为非负数）
    result = linprog(-c, A_eq=A_eq, b_eq=b_eq, bounds=bounds)
    if result.success:
        print("Optimal Solution Found!")
        quantities = result.x
        for i, code in enumerate(product_codes):
            print(f"Product {code}: Quantity = {quantities[i]:.2f}")
    else:
        print("No Optimal Solution Found.")
    import numpy as np
    import pandas as pd
    import matplotlib.pyplot as plt
    from scipy.optimize import linprog
```

```
plt.rcParams['font.sans-serif'] = ['SimHei']
product_codes = ["花叶类", "花菜类", "水生根茎类", "茄类", "辣椒类", "食用菌"]
prices = [20, 8, 19, 12, 7, 6]
costs = [6, 3, 4, 5, 7, 9]
min_display = [2.5, 2.5, 2.5, 2.5, 2.5, 2.5]
c = np.array(prices) - np.array(costs) # 目标函数系数（最大化总收益）
A_eq = np.ones((1, len(product_codes))) # 约束矩阵（总售卖单品数量）
b_eq = np.array([27]) # 约束条件（总售卖单品数量限制在 27-33 个之间）
bounds = [(0, None) for _ in product_codes] # 变量边界（补货量为非负数）
result = linprog(-c, A_eq=A_eq, b_eq=b_eq, bounds=bounds)
if result.success:
    print("Optimal Solution Found!")
    quantities = result.x
    for i, code in enumerate(product_codes):
        print(f"Product {code}: Quantity = {quantities[i]:.2f}")
else:
    print("No Optimal Solution Found.")
result_df = pd.DataFrame({"Product Code": product_codes, "Quantity": quantities})
result_df.to_excel("G:/mathematic/cvs/result.xlsx", index=False)
plt.bar(product_codes, quantities)
plt.xlabel("分类名称")
plt.ylabel("进货量")
plt.title("最优补货策略")
plt.savefig("G:/mathematic/result_plot.png")
plt.show()
```

画图.py

```
import pandas as pd

import matplotlib.pyplot as plt

from matplotlib.font_manager import FontProperties

import seaborn as sns

merged_table = pd.read_excel(r'D:\\文件\\数学建模\\2023 数模\\merge.xlsx')

vegetable_sales = merged_table.groupby('分类名称')['销量(千克)'].sum().reset_index()

result_df = pd.DataFrame(vegetable_sales, columns=['分类名称', '销量(千克)'])

print(result_df)

import seaborn as sns

import matplotlib.pyplot as plt

import pylab as mpl

result_df_sorted = result_df.sort_values(by='销量(千克)', ascending=False)

plt.figure(figsize=(10, 6))

sns.barplot(x='分类名称', y='销量(千克)', data=result_df_sorted, palette='Set2', ci=None)

plt.grid(axis='x')

plt.tight_layout()

plt.xticks(rotation=45)

plt.show()

monthly_sales = merged_table.set_index('销售日期').groupby('分类名称')['销量(千
克)'].resample('M').sum().reset_index()

result_df = pd.DataFrame(monthly_sales)

print(result_df)

import matplotlib.pyplot as plt

vegetable_name = '茄类'
```

```
vegetable_data = result_df[result_df['分类名称'] == vegetable_name]

plt.figure(figsize=(10, 6))

plt.plot(vegetable_data['销售日期'], vegetable_data['销量(千克)'], marker='o', linestyle='-')

plt.title(f'{vegetable_name} 月度销量统计')

plt.xlabel('月份')

plt.ylabel('销量(千克)')

plt.grid(True)

plt.show()

vegetable_categories = merged_table['分类名称'].unique()

for category in vegetable_categories:

    vegetable_data = result_df[result_df['分类名称'] == category]

    plt.plot(vegetable_data['销售日期'], vegetable_data['销量(千克)'], label=category, marker='o', linestyle='-')

plt.title('各种蔬菜月度销量统计')

plt.xlabel('月份')

plt.ylabel('销量(千克)')

plt.grid(True)

plt.show()

import pandas as pd

monthly_sales_grouped = merged_table.groupby(['分类名称', pd.Grouper(key='销售日期', freq='M')])['销量(千克)'].sum().reset_index()

monthly_sales_dict = {}

for category in monthly_sales_grouped['分类名称'].unique():

    category_data = monthly_sales_grouped[monthly_sales_grouped['分类名称'] == category]

    monthly_sales_list = category_data['销量(千克)'].tolist()

    monthly_sales_dict[category] = monthly_sales_list
```

```
print(monthly_sales_dict)

import pandas as pd

max_length = max(len(values) for values in monthly_sales_dict.values())

filled_monthly_sales_dict = {category: values + [0] * (max_length - len(values)) for category,
values in monthly_sales_dict.items()}

sales_df = pd.DataFrame(filled_monthly_sales_dict)

monthly_sales_df = monthly_sales_df.fillna(0)

correlation_matrix = sales_df.corr()

print(correlation_matrix)

import seaborn as sns

import matplotlib.pyplot as plt

plt.figure(figsize=(10, 8))

sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")

plt.title('蔬菜销量相关性热力图')

plt.show()

element_counts = {}

for vegetable, sales_data in monthly_sales_dict.items():

    element_count = len(sales_data)

    element_counts[vegetable] = element_count

for vegetable, count in element_counts.items():

    print(f"{vegetable} 销量列表有 {count} 个元素。")

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt
```

```
daily_sales_grouped = merged_table.groupby(['分类名称', pd.Grouper(key='销售日期', freq='D')])['销量(千克)'].sum().reset_index()

vegetable_categories = daily_sales_grouped['分类名称'].unique()

daily_sales_dict = {}

max_length = daily_sales_grouped.groupby('分类名称')['销量(千克)'].transform('count').max()

for category in vegetable_categories:

    category_data = daily_sales_grouped[daily_sales_grouped['分类名称'] == category]

    daily_sales_list = category_data.set_index('销售日期')['销量(千克)'].resample('D').sum().fillna(0).tolist()

    daily_sales_list += [0] * (max_length - len(daily_sales_list))

    daily_sales_dict[category] = daily_sales_list

sales_df = pd.DataFrame(daily_sales_dict)

correlation_matrix = sales_df.corr()

print(correlation_matrix)

plt.figure(figsize=(12, 8))

sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")

plt.title('蔬菜销量相关性热力图')

plt.xlabel('蔬菜分类')

plt.ylabel('蔬菜分类')

plt.xticks(rotation=45)

plt.show()

import pandas as pd

import matplotlib.pyplot as plt

import re

def extract_vegetable_name(name):

    return re.sub(r'\(\d+\)', '', name)
```

```
merged_table['基本蔬菜名称'] = merged_table['分类名称'].apply(extract_vegetable_name)

vegetable_frequency = merged_table['基本蔬菜名称'].value_counts()

print(vegetable_frequency)

plt.figure(figsize=(12, 8))

vegetable_frequency.plot(kind='bar')

plt.title('各个单品销售频数统计')

plt.xlabel('蔬菜名称')

plt.ylabel('销售频数')

plt.xticks(rotation=45)

plt.show()

import pandas as pd

from sklearn.preprocessing import StandardScaler

from scipy.cluster.hierarchy import dendrogram, linkage

import matplotlib.pyplot as plt

import seaborn as sns

merged_table['处理后的单品名称'] = merged_table['单品名称'].str.replace(r'\(\d+\)', '', regex=True)

grouped_data = merged_table.groupby(['处理后的单品名称', pd.Grouper(key='销售日期', freq='D')])['销量(千克)'].sum().reset_index()

print(grouped_data)

import pandas as pd

from sklearn.preprocessing import StandardScaler

from scipy.cluster.hierarchy import dendrogram, linkage

import matplotlib.pyplot as plt

from matplotlib.font_manager import FontProperties

merged_table['处理后的单品名称'] = merged_table['单品名称'].str.replace(r'\(\d+\)', '', regex=True)
```

```
merged_table['处理后的单品名称'] = merged_table['处理后的单品名称'].str.replace(r'\(.*\)', '', regex=True)

grouped_data = merged_table.groupby(['处理后的单品名称'])['销量(千克)'].sum().reset_index()

scaler = StandardScaler()

scaled_data = scaler.fit_transform(grouped_data[['销量(千克)']])

linkage_matrix = linkage(scaled_data, method='ward', metric='euclidean')

plt.figure(figsize=(8, 48))

dendrogram(linkage_matrix, labels=grouped_data['处理后的单品名称'].tolist(), orientation='right', leaf_rotation=0, leaf_font_size=12)

plt.title('层次聚类树状图', fontproperties=font,fontsize=16)

plt.xlabel('距离', size=14,fontproperties=font)

plt.ylabel('蔬菜单品名称',size=14, fontproperties=font)

plt.savefig('cluster_dendrogram.png', dpi=300)

plt.show()

import pandas as pd

from sklearn.preprocessing import StandardScaler

from sklearn.cluster import KMeans

import matplotlib.pyplot as plt

import seaborn as sns

grouped_data = merged_table.groupby(['单品名称'])['销量(千克)'].sum().reset_index()

print(grouped_data)

price_data = pd.read_excel(r'D:\文件\数学建模\2023 数模\附件 44.xlsx')

print(price_data)


import pandas as pd

from sklearn.preprocessing import StandardScaler
```

```
grouped_data = pd.merge(grouped_data, price_data, on='单品名称')
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(scaled_data)
    inertia_values.append(kmeans.inertia_)
plt.figure(figsize=(8, 6))
plt.plot(range(1, 21), inertia_values, marker='o', linestyle='-', color='b')
plt.xlabel('K 值')
plt.ylabel('簇内平方和')
plt.title('肘部法则')
plt.xticks(range(1, 21))
plt.grid(True)
plt.show()
kmeans = KMeans(n_clusters=best_k, random_state=42)
grouped_data['Cluster'] = cluster_labels
plt.figure(figsize=(12, 8))
sns.scatterplot(x='损耗率(%)', y='销量(千克)', hue='Cluster', data=grouped_data, palette='viridis')
plt.title('蔬菜销量 K 均值聚类散点图')
plt.xlabel('损耗率(%)')
plt.ylabel('销量(千克)')
plt.show()
grouped_data = merged_table.groupby(['单品名称'])['销量(千克)'].sum().reset_index()
grouped_data['单品编码'] = merged_table.groupby(['单品名称'])['单品编码'].first().values
print(grouped_data)
price_data = pd.read_excel(r'D:\文件\数学建模\2023 数模\附件 3.xlsx')
average_prices = price_data.groupby('单品编码')['批发价格'].mean().reset_index()
```

```
print(average_prices)
grouped_data = pd.merge(grouped_data, average_prices, on='单品编码')
print(grouped_data)
scaler = StandardScaler()
scaled_data = scaler.fit_transform(grouped_data[['销量(千克)', '批发价格']])
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(scaled_data)
    inertia_values.append(kmeans.inertia_)
plt.figure(figsize=(8, 6))
plt.plot(range(1, 21), inertia_values, marker='o', linestyle='-', color='b')
plt.xlabel('K 值')
plt.ylabel('簇内平方和')
plt.title('肘部法则')
plt.xticks(range(1, 21))
plt.grid(True)
plt.show()
kmeans = KMeans(n_clusters=best_k, random_state=42)
grouped_data['Cluster'] = cluster_labels
plt.figure(figsize=(12, 8))
sns.scatterplot(x='批发价格', y='销量(千克)', hue='Cluster', data=grouped_data, palette='viridis')
plt.title('蔬菜销量 K 均值聚类散点图')
plt.xlabel('批发价格(元/千克)')
plt.ylabel('销量(千克)')
plt.legend(title='Cluster', loc='upper right')
plt.show()
kmeans = KMeans(n_clusters=best_k, random_state=42)
```

```
grouped_data['Cluster'] = cluster_labels

plt.figure(figsize=(12, 8))

sns.scatterplot(x='批发价格(元/千克)', y='销量(千克)', hue='Cluster', data=grouped_data, palette='viridis')

plt.title('蔬菜销量 K 均值聚类散点图')

plt.xlabel('批发价格(元/千克)')

plt.ylabel('销量(千克)')

plt.legend(title='Cluster', loc='upper right')

plt.show()

import pandas as pd

grouped_data = merged_table.groupby(['单品名称'])['销量(千克)'].sum().reset_index()

average_price = merged_table.groupby(['单品名称'])['销售单价(元/千克)'].mean().reset_index()

result = pd.merge(grouped_data, average_price, on='单品名称')

print(result)

scaler = StandardScaler()

scaled_data = scaler.fit_transform(result[['销量(千克)', '销售单价(元/千克)']])

non_numeric_values = result[~result['销售单价(元/千克)'].apply(lambda x: str(x).replace('.', '', 1).isdigit())]

print(non_numeric_values)

print(merged_table['销售单价(元/千克)'].dtype)


merged_table['销售单价(元/千克)'] = pd.to_numeric(merged_table['销售单价(元/千克)'], errors='coerce')

    kmeans = KMeans(n_clusters=k, random_state=42)

    kmeans.fit(scaled_data)

    inertia_values.append(kmeans.inertia_)

plt.figure(figsize=(8, 6))
```

```
plt.plot(range(1, 21), inertia_values, marker='o', linestyle='-', color='b')

plt.xlabel('K 值')

plt.ylabel('簇内平方和')

plt.title('肘部法则')

plt.xticks(range(1, 21))

plt.grid(True)

plt.show()

kmeans = KMeans(n_clusters=best_k, random_state=42)

grouped_data['Cluster'] = cluster_labels

plt.figure(figsize=(12, 8))

sns.scatterplot(x='销售单价(元/千克)', y='销量(千克)', hue='Cluster', data=grouped_data,
palette='viridis')

plt.title('蔬菜销量 K 均值聚类散点图')

plt.xlabel('销售单价(元/千克)')

plt.ylabel('销量(千克)')

plt.legend(title='Cluster', loc='upper right')

plt.show()
```