

النسخة النهائية والعملية من الكود

يتضمن هذا الكود جميع الخطوات، بما في ذلك قراءة البيانات، واختيار الميزات بطريقتي RFE و Chi-Square ، وتدريب نماذج Random Forest مع تحسين المعلمات، بالإضافة إلى عرض النتائج والجدول والمخططات البيانية المطلوبة.

كيفية الاستخدام:

1. **التنزيل**: يمكنك نسخ الكود من الـ Canvas أعلاه وحفظه في ملف بامتداد (ml_pipeline.py).مثلاً
2. **ملف البيانات**: تأكد من وجود ملف البيانات kidney_disease_clean_final.csv في نفس المجلد الذي يوجد به ملف البايثون، أو قم بتعديل المسار في متغير data_file_path داخل الكود.
3. **المكتبات**: تأكد من تثبيت جميع المكتبات المطلوبة إذا لم تكن مثبتة بالفعل :
Bash
pip install pandas numpy matplotlib seaborn scikit-learn ipython
(مكتبة python اختيارية ولكنها تحسن عرض الجداول في بعض البيئات.)
4. **التشغيل**: افتح الطرفية (Terminal) أو موجه الأوامر (Command Prompt) ، انتقل إلى المجلد الذي حفظت فيه الملف، وقم بتشغيل السكريبت باستخدام الأمر :
Bash
python ml_pipeline.py
5. **النتائج**: سيُقوم السكريبت بطباعة خطوات التنفيذ، والجدول، والنتائج، وعرض المخططات البيانية في نوافذ منفصلة.

```
# -*- coding: utf-8 -*-
"""
. مشروع تطبيق نماذج التعلم الآلي مع اختيار الميزات وتحسين المعلمات الفائقة

يهدف هذا السكريبت إلى تنفيذ الخطوات التالية:
1. CSV. قراءة البيانات من ملف
2. تقسيم البيانات إلى مجموعات تدريب واختبار
3. لاختيار الميزات وعرض النتائج بيانياً Chi-Square تطبيق طريقة
4. لاختيار الميزات Recursive Feature Elimination (RFE) تطبيق طريقة
5. GridSearchCV. مع تحسين Chi-Square باستخدام الميزات المختارة بواسطة Random Forest تدريب وتقييم نموذج
6. GridSearchCV. مع تحسين RFE باستخدام الميزات المختارة بواسطة Random Forest تدريب وتقييم نموذج
7. عرض مصفوفات الارتباك وأهمية الميزات بيانياً

المنفذون: [أضف أسماء المنفذين هنا]
التاريخ: [أضف تاريخ التنفيذ هنا]
"""

# استيراد المكتبات اللازمة
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from typing import Tuple, Optional, List, Dict, Any

# مكتبات Scikit-learn
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.feature_selection import SelectKBest, chi2, RFE
from sklearn.linear_model import LogisticRegression # يستخدم كمقدر في RFE
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import (
    accuracy_score,
    precision_score,
    recall_score,
    f1_score,
    classification_report,
    confusion_matrix,
)
from sklearn.exceptions import NotFittedError

# --- إعدادات العرض للرسوم البيانية ---
# استخدام نمط مرئي جذاب للرسوم
sns.set_theme(style="whitegrid", palette="pastel")
# التأكد من دعم الخطوط العربية (قد تحتاج لتثبيت خطوط إضافية أو تغيير اسم الخط حسب نظام التشغيل)
# أمثلة لخطوط قد تدعم العربية: 'Arial', 'Tahoma', 'DejaVu Sans'
plt.rcParams['font.family'] = 'DejaVu Sans'
# تحديد حجم افتراضي للرسوم لسهولة القراءة
plt.rcParams['figure.figsize'] = (12, 7) # زيادة طفيفة في الحجم الافتراضي
plt.rcParams['figure.dpi'] = 90 # تحسين دقة الرسوم قليلاً

# =====
# 1. قراءة البيانات وتقسيمها
# =====
def load_and_split_data(
    data_file_path: str, target_column: str = "classification", test_size: float = 0.2, random_state: int = 42
) -> Optional[Tuple[pd.DataFrame, pd.DataFrame, pd.Series, pd.Series]]:
    """
    . وتقسيمها إلى مجموعات تدريب واختبار CSV تحميل البيانات من ملف

    المدخلات:
    data_file_path (str): الذي يحتوي على البيانات CSV مسار ملف
    target_column (str): اسم العمود الذي يمثل المتغير الهدف. الافتراضي هو 'classification'.
    test_size (float): نسبة البيانات التي ستستخدم كمجموعة اختبار. الافتراضي هو 0.2 (20%).
    random_state (int): قيمة البذرة العشوائية لضمان قابلية تكرار النتائج. الافتراضي هو 42.

    المخرجات:
    Optional[Tuple[pd.DataFrame, pd.DataFrame, pd.Series, pd.Series]]:
        (X_train, X_test, y_train, y_test) التحميل والتقسيم بنجاح.
        None إذا حدث خطأ
    """
    print("=====")
    print("الخطوة 1: قراءة البيانات وتقسيمها")
    print("=====")

    try:
        # قراءة مجموعة البيانات
        data = pd.read_csv(data_file_path)
        print(f"تم قراءة ملف البيانات بنجاح من {data_file_path}")
        print(f"أبعاد البيانات الأصلية: {data.shape}")

        # عرض معلومات أساسية عن البيانات
        print("\nمعلومات عن البيانات (الأنواع والقيم غير الفارغة):")
        data.info()

        # عرض إحصائيات وصفية للميزات الرقمية
```

```

print("\nإحصائيات وصفية للميزات الرقمية:")
try:
    from IPython.display import display
    display(data.describe().T) # .T لقالب الجدول لمسهولة القراءة
except ImportError:
    print(data.describe().T.to_string())

# عرض أول 5 صفوف من البيانات للتأكد
print("\nأول 5 صفوف من البيانات:")
try:
    from IPython.display import display
    display(data.head())
except ImportError:
    print(data.head().to_string())

# التحقق من وجود العمود الهدف
if target_column not in data.columns:
    print(f"\nخطأ: '{target_column}' غير موجود في ملف البيانات.")
    print("يرجى التأكد من اسم عمود المتغير الهدف وتمثيله للدالة إذا كان مختلفاً.")
    return None

# فصل الميزات (y) عن المتغير الهدف (X)
X = data.drop(target_column, axis=1)
y = data[target_column]

# عرض توزيع الفئات في المتغير الهدف
print(f"\nتوزيع الفئات في المتغير الهدف: '{target_column}':")
print(y.value_counts(normalize=True) * 100) # عرض النسب المئوية

# تقسيم مجموعة البيانات إلى مجموعتي تدريب واختبار
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=test_size, random_state=random_state, stratify=y # الحفاظ على نسبة الفئات stratify استخدام
)

print("\nتم تقسيم البيانات بنجاح.")
print(f"\nعدد الأمثلة في مجموعة التدريب: {X_train.shape[0]} فئة {y.unique()[0]}, {len(y_train[y_train == y.unique()[1]])} فئة {y.unique()[1]}")
print(f"\nعدد الأمثلة في مجموعة الاختبار: {X_test.shape[0]} فئة {y.unique()[0]}, {len(y_test[y_test == y.unique()[1]])} فئة {y.unique()[1]}")
print(f"\nعدد الميزات الأصلية: {X_train.shape[1]}")

# التحقق من عدم وجود قيم مفقودة (NaN)
missing_train = X_train.isnull().sum().sum()
missing_test = X_test.isnull().sum().sum()
if missing_train > 0 or missing_test > 0:
    print(f"\n{missing_train} قيم مفقودة في مجموعة التدريب و {missing_test} قيم مفقودة في مجموعة الاختبار.")
    print("قبل المتابعة (SimpleImputer) مثل الحذف أو الإكمال باستخدام) قد نحتاج إلى معالجة القيم المفقودة")
    # مثال على الإكمال باستخدام المتوسط (للميزات الرقمية فقط)
    # from sklearn.impute import SimpleImputer
    # imputer = SimpleImputer(strategy='mean')
    # X_train = pd.DataFrame(imputer.fit_transform(X_train), columns=X_train.columns, index=X_train.index)
    # X_test = pd.DataFrame(imputer.transform(X_test), columns=X_test.columns, index=X_test.index)
    # print("تم إكمال القيم المفقودة باستخدام المتوسط")
else:
    print("\nلا توجد قيم مفقودة في مجموعتي التدريب والاختبار")

return X_train, X_test, y_train, y_test

except FileNotFoundError:
    print(f"\nخطأ: ملف البيانات '{data_file_path}' غير موجود.")
    print("يرجى التأكد من وجود الملف في المسار الصحيح أو تعديل المتغير 'data_file_path'.")
    return None
except pd.errors.EmptyDataError:
    print(f"\nخطأ: ملف البيانات '{data_file_path}' فارغ.")
    return None
except Exception as e:
    print(f"\nحدث خطأ غير متوقع أثناء قراءة أو تقسيم البيانات: {e}")
    import traceback
    traceback.print_exc() # طباعة تتبع الخطأ للمساعدة في التشخيص
    return None

# =====
# وعرض النتائج (Chi-Square) اختيار الميزات باستخدام طريقة الفلتر 2.
# =====
def select_features_chi2(
    X_train: pd.DataFrame, y_train: pd.Series, X_test: pd.DataFrame, n_features_to_select: int
) -> Optional[Tuple[np.ndarray, np.ndarray, pd.DataFrame, List[str]]]:
    """
    وعرض النتائج بيانياً Chi-Square اختيار أفضل الميزات باستخدام اختبار

    المدخلات:
    X_train (DataFrame): ميزات مجموعة بيانات التدريب.
    y_train (Series): المتغير الهدف لمجموعة التدريب.
    X_test (DataFrame): ميزات مجموعة بيانات الاختبار.
    n_features_to_select (int): عدد الميزات المراد اختيارها.

    المخرجات:
    Optional[Tuple[np.ndarray, np.ndarray, pd.DataFrame, List[str]]]:
        إذا نجح الاختبار (X_train_selected, X_test_selected, chi2_scores_df_top, selected_features_names)
        X_train_selected: مصفوفة numpy لبيانات التدريب بعد اختيار الميزات.
        X_test_selected: مصفوفة numpy لبيانات الاختبار بعد اختيار الميزات.
        chi2_scores_df_top: DataFrame يحتوي على درجات Chi2 لأفضل الميزات المختارة.
        selected_features_names: قائمة بأسماء أفضل الميزات المختارة.
        None إذا حدث خطأ.
    """
    print("\n=====")
    print("الخطوة 2: اختيار الميزات باستخدام طريقة الفلتر (Chi-Square)")
    print("=====")

    original_feature_names = X_train.columns.tolist()

    try:

```

```

# --- التحقق من القيم السالبة ---
negative_values = X_train < 0
if negative_values.any().any():
    print("\nيتطلب قيمًا غير سالبة Chi-Square. تحذير: تم العثور على قيم سالبة في بيانات التدريب.")
    negative_cols = X_train.columns[negative_values.any()].tolist()
    print(f"الأعمدة التي تحتوي على قيم سالبة: {negative_cols}")
    print("Chi-Square (مثل MinMaxScaler) قد لا يكون مناسبًا لهذه البيانات أو قد تحتاج إلى تحويلها.")
    # يمكنك اختيار إيقاف التنفيذ هنا أو المتابعة مع تحذير
    # return None # لإيقاف التنفيذ
    # تطبيق MinMaxScaler أو تطبيق تحويل مثل
    from sklearn.preprocessing import MinMaxScaler
    scaler = MinMaxScaler()
    X_train_scaled = scaler.fit_transform(X_train)
    X_test_scaled = scaler.transform(X_test)
    # للحفاظ على أسماء الأعمدة DataFrame تحويل المصفوفات الناتجة مرة أخرى إلى
    X_train = pd.DataFrame(X_train_scaled, columns=original_feature_names, index=X_train.index)
    X_test = pd.DataFrame(X_test_scaled, columns=original_feature_names, index=X_test.index)
    print("[0, 1]. لتحويل البيانات إلى النطاق MinMaxScaler تم تطبيق")

# --- SelectKBest تهيئة وتطبيق ---
# للحصول على درجات جميع الميزات أولاً k='all' استخدام
selector_chi2 = SelectKBest(chi2, k='all')
selector_chi2.fit(X_train, y_train)

# --- الحصول على الدرجات والترتيب ---
chi2_scores = selector_chi2.scores_
# التي قد تنتج إذا كانت الميزة ثابتة أو غير مفيدة NaN التعامل مع القيم اللانهائية أو
chi2_scores = np.nan_to_num(chi2_scores, nan=0.0, posinf=np.finfo(np.float64).max, neginf=-np.finfo(np.float64).max)

chi2_scores_df = pd.DataFrame({
    "Feature": original_feature_names,
    "Chi2 Score": chi2_scores
})
# إذا كانت مفيدة للتحليل P (p-values) إضافة قيم
# chi2_scores_df['P-value'] = selector_chi2.pvalues_
chi2_scores_df = chi2_scores_df.sort_values(by="Chi2 Score", ascending=False).reset_index(drop=True)

# --- التحقق من عدد الميزات المطلوب ---
num_available_features = len(original_feature_names)
if n_features_to_select > num_available_features:
    print(f"أكبر من عدد الميزات المتاحة ({num_available_features}). تحذير: عدد الميزات المطلوب.")
    print(f"مييزات {num_available_features} سيتم اختيار أفضل")
    n_features_to_select = num_available_features
elif n_features_to_select <= 0:
    print("خطأ: يجب أن يكون عدد الميزات المطلوب أكبر من صفر")
    return None

# --- اختيار أفضل الميزات وعرضها ---
chi2_scores_df_top = chi2_scores_df.head(n_features_to_select)
selected_features_names = chi2_scores_df_top['Feature'].tolist()

print(f"Chi-Square ميزات تم تحديدها باستخدام أفضل {n_features_to_select} ودرجاتها:")
try:
    from IPython.display import display
    display(chi2_scores_df_top)
except ImportError:
    print(chi2_scores_df_top.to_string())

# --- Chi-Square رسم بياني لدرجات ---
plt.figure(figsize=(10, max(5, n_features_to_select * 0.6))) # تعديل الحجم ديناميكيًا
barplot = sns.barplot(x="Chi2 Score", y="Feature", data=chi2_scores_df_top,
    palette="viridis", hue="Feature", dodge=False, legend=False) # إعطاء كل شريط لون مميز hue استخدام
plt.title(f'Chi-Square ميزات حسب درجة {n_features_to_select} أعلى')
plt.xlabel('Chi-Square درجة', fontsize=14)
plt.ylabel('الميزة', fontsize=14)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
# إضافة قيم الدرجات على الأعمدة لتوضيح أكبر
for index, value in enumerate(chi2_scores_df_top['Chi2 Score']):
    plt.text(value, index, f'{value:.2f}', va='center', ha='left', fontsize=10, color='black') # تعديل ha و fontsize
plt.tight_layout()
plt.show()
# -----

# --- SelectKBest إعادة تهيئة ---
# ولكن نغير قيمة k ونستخدم نفس الكائن selector_chi2
final_selector_chi2 = SelectKBest(chi2, k=n_features_to_select)
# ندرّب المحول النهائي على بيانات التدريب (المُخَمَّجة إذا تم التحجيم)
final_selector_chi2.fit(X_train, y_train)
# نحصل على القناع البولياني للميزات المختارة
support_mask = final_selector_chi2.get_support()
# نحصل على أسماء الميزات المختارة فعليًا بواسطة المحول
selected_features_names_from_selector = X_train.columns[support_mask].tolist()

# تحويل مجموعتي التدريب والاختبار
X_train_selected = final_selector_chi2.transform(X_train)
X_test_selected = final_selector_chi2.transform(X_test)

print("\nبنجاح Chi-Square تم اختيار الميزات باستخدام")
print(f"{X_train_selected.shape}: الشكل الجديد لمجموعة التدريب بعد اختيار الميزات")
print(f"{X_test_selected.shape}: الشكل الجديد لمجموعة الاختبار بعد اختيار الميزات")
print(f"{selected_features_names_from_selector}: أسماء الميزات المختارة فعليًا بواسطة المحول")

# التأكد من أن الأسماء التي تم فرزها هي نفسها التي تم اختيارها
if set(selected_features_names) != set(selected_features_names_from_selector):
    print("SelectKBest. تحذير: قد يكون هناك اختلاف بسيط في مجموعة الميزات المختارة بسبب تساوي الدرجات أو طريقة عمل")
    print(f"{selected_features_names}: الميزات المختارة بناءً على الفرز")
    selected_features_names = selected_features_names_from_selector # استخدام الأسماء من المحول لضمان التوافق مع البيانات المحولة

return X_train_selected, X_test_selected, chi2_scores_df_top, selected_features_names

except ValueError as e:

```

```

print(f"\nحدث ValueError أثناء تطبيق Chi-Square: {e}")
print("قد يكون السبب وجود قيم سالبة في البيانات (إذا لم يتم التعامل معها) أو عدم ملائمة نوع البيانات")
import traceback
traceback.print_exc()
return None
except NotFittedError:
    print("\nNotFittedError: محاولة استخدام المحول قبل تدريبه (fit).")
    import traceback
    traceback.print_exc()
    return None
except Exception as e:
    print(f"\nحدث خطأ غير متوقع أثناء اختيار الميزات: {e}")
    import traceback
    traceback.print_exc()
    return None

# =====
# 3. اختيار الميزات باستخدام طريقة المغلف (RFE)
# =====
def select_features_rfe(
    X_train: pd.DataFrame, y_train: pd.Series, X_test: pd.DataFrame, n_features_to_select: int,
    # خيار استخدام مقدر بسيط وقوي مثل
    # estimator = LogisticRegression(max_iter=2000, random_state=42, solver='liblinear', class_weight='balanced')
    estimator = RandomForestClassifier(n_estimators=50, random_state=42, class_weight='balanced', n_jobs=-1) # RF قد يكون أبطأ ولكنه قد يلتقط علاقات غير خطية
) -> Optional[Tuple[np.ndarray, np.ndarray, List[str]]]:
    """
    Recursive Feature Elimination (RFE). اختيار أفضل الميزات باستخدام

    المدخلات:
    X_train (DataFrame): ميزات مجموعة بيانات التدريب.
    y_train (Series): المتغير الهدف لمجموعة التدريب.
    X_test (DataFrame): ميزات مجموعة بيانات الاختيار.
    n_features_to_select (int): عدد الميزات المراد اختيارها.
    estimator (object): نموذج التقدير المستخدم داخل RFE. الافتراضي هو RandomForestClassifier.

    المخرجات:
    Optional[Tuple[np.ndarray, np.ndarray, List[str]]]:
        إذا نجح الاختيار (X_train_selected, X_test_selected, selected_features_names)
        لبيانات التدريب بعد اختيار الميزات numpy مصفوفة.
        لبيانات الاختيار بعد اختيار الميزات numpy مصفوفة.
        قائمة بأسماء الميزات المختارة selected_features_names.
        إذا حدث خطأ None.
    """
    print("\n\n=====")
    print(f"\nمع مقدر {type(estimator).__name__} (RFE) الخطوة 3: اختيار الميزات باستخدام طريقة المغلف")
    print("=====")

    original_feature_names = X_train.columns.tolist()

    # --- التحقق من عدد الميزات المطلوب ---
    num_available_features = len(original_feature_names)
    if n_features_to_select > num_available_features:
        print(f"\nتحذير: عدد الميزات المطلوب ({n_features_to_select}) أكبر من عدد الميزات المتاحة ({num_available_features}).")
        print(f"ميزات {num_available_features} سيتم اختيار أفضل")
        n_features_to_select = num_available_features
    elif n_features_to_select <= 0:
        print("\nخطأ: يجب أن يكون عدد الميزات المطلوب أكبر من صفر")
        return None

    try:
        # --- RFE تهيئة ---
        # step=0.1 يعني إزالة 10% من الميزات في كل خطوة (قد يكون أسرع للمجموعات الكبيرة)
        # step=1 يعني إزالة ميزة واحدة في كل خطوة (أكثر دقة ولكن أبطأ)
        rfe_selector = RFE(
            estimator=estimator, n_features_to_select=n_features_to_select, step=1, verbose=0
        )

        print(f"\n...ميزات {n_features_to_select} لاختيار أفضل RFE تطبيق")
        # --- تطبيق RFE ---
        rfe_selector.fit(X_train, y_train) # RFE يتطلب تدريب المقدر داخليًا

        # --- الحصول على الميزات المختارة ---
        support_mask = rfe_selector.support_ # قناع بولياني للميزات المختارة
        selected_features_names = X_train.columns[support_mask].tolist()

        print(f"\nRFE (الميزات المختارة باستخدام RFE ({len(selected_features_names)} ميزات):")
        print(selected_features_names)

        # --- عرض ترتيب الميزات (اختياري) ---
        if hasattr(rfe_selector, 'ranking_'):
            feature_ranking = pd.DataFrame({
                'Feature': original_feature_names,
                'Selected': support_mask,
                'Ranking': rfe_selector.ranking_ # 1 الميزة المختارة
            }).sort_values(by='Ranking')
            print(f"\nRFE ترتيب الميزات حسب (1 المختار):")
            try:
                from IPython.display import display
                display(feature_ranking)
            except ImportError:
                print(feature_ranking.to_string())
        # -----

        # --- تحويل البيانات ---
        X_train_selected = rfe_selector.transform(X_train)
        X_test_selected = rfe_selector.transform(X_test)

        print("\nRFE تم اختيار الميزات باستخدام")
        print(f"الشكل الجديد لمجموعة التدريب بعد اختيار الميزات: {X_train_selected.shape}")
        print(f"الشكل الجديد لمجموعة الاختيار بعد اختيار الميزات: {X_test_selected.shape}")

        return X_train_selected, X_test_selected, selected_features_names

```

```

except NotFittedError:
    print("\nNotFittedError: محاولة استخدام المحول RFE قبل تدريبه (fit).")
    import traceback
    traceback.print_exc()
    return None
except Exception as e:
    print(f"\nحدث خطأ غير متوقع أثناء اختيار الميزات بـRFE: {e}")
    import traceback
    traceback.print_exc()
    return None

# =====
# دالة مساعدة لتقييم النموذج 4.
# =====
def evaluate_model(model_name: str, y_true: np.ndarray, y_pred: np.ndarray) -> Dict[str, Any]:
    """
    تقييم النموذج وحساب مقاييس الأداء المختلفة.

    المدخلات:
    model_name (str): اسم النموذج لوصف النتائج.
    y_true (array-like): القيم الحقيقية للمتغير الهدف.
    y_pred (array-like): القيم المتوقعة من النموذج.

    المخرجات:
    Dict[str, Any]: قاموس يحتوي على اسم النموذج ومقاييس الأداء المحسوبة.
    """
    try:
        accuracy = accuracy_score(y_true, y_pred)
        # لحساب المتوسط الموزون للمقاييس، وهو جيد للبيانات غير المتوازنة 'average='weighted' استخدام
        precision = precision_score(y_true, y_pred, average="weighted", zero_division=0)
        recall = recall_score(y_true, y_pred, average="weighted", zero_division=0)
        f1 = f1_score(y_true, y_pred, average="weighted", zero_division=0)
        # الحصول على تقرير التصنيف كخمس لمهولة العرض
        report = classification_report(y_true, y_pred, zero_division=0, output_dict=False)
        # حساب مصفوفة الارتباك
        cm = confusion_matrix(y_true, y_pred)
        # confusion matrix labels لـ y_true أو y_pred الحصول على أسماء الفئات الفريدة من
        # التأكد من أن الترتيب متسق
        labels = sorted(list(np.unique(np.concatenate((y_true, y_pred)))))

        metrics = {
            "Model Name": model_name,
            "Accuracy": accuracy,
            "Precision (Weighted)": precision,
            "Recall (Weighted)": recall,
            "F1 Score (Weighted)": f1,
            "Classification Report": report,
            "Confusion Matrix": cm,
            "Labels": labels # لإستخدامها في الرسم labels إضافة إلى
        }
        return metrics
    except Exception as e:
        print(f"\nحدث خطأ أثناء تقييم النموذج '{model_name}': {e}")
        import traceback
        traceback.print_exc()
        return {"Model Name": model_name, "Error": str(e)} # إرجاع خطأ في حالة الفشل

# =====
# دالة مساعدة لطباعة نتائج التقييم ورسم مصفوفة الارتباك 5.
# =====
def print_evaluation_results(evaluation_results: Dict[str, Any]):
    """
    طباعة نتائج التقييم بشكل منسق ورسم مصفوفة الارتباك.

    المدخلات:
    evaluation_results (Dict[str, Any]): القاموس الناتج من دالة evaluate_model.
    """
    if not evaluation_results or "Model Name" not in evaluation_results:
        print("لا توجد نتائج تقييم لعرضها")
        return

    model_name = evaluation_results.get('Model Name', 'N/A')
    print(f"\n--- نتائج تقييم: {model_name} ---")
    if "Error" in evaluation_results:
        print(f"خطأ في التقييم: {evaluation_results['Error']}")
        return

    # طباعة المقاييس الرقمية أولاً
    print("\nالمقاييس الرئيسية:")
    for key, value in evaluation_results.items():
        if key in ["Model Name", "Classification Report", "Confusion Matrix", "Labels"]:
            continue
        if isinstance(value, float):
            print(f"    {key}: {value:.4f}") # تنسيق الأرقام العشرية
        else:
            print(f"    {key}: {value}") # لطباعة أي مقاييس أخرى غير عشرية

    # طباعة تقرير التصنيف
    if "Classification Report" in evaluation_results:
        print("\n    تقرير التصنيف التفصيلي")
        print(evaluation_results["Classification Report"])

    # --- رسم مصفوفة الارتباك ---
    if "Confusion Matrix" in evaluation_results and "Labels" in evaluation_results:
        cm = evaluation_results["Confusion Matrix"]
        labels = evaluation_results["Labels"]
        # تعديل الحجم قليلاً
        plt.figure(figsize=(7, 6))
        sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', # لإظهار الأرقام
                    # لإضافة خطوط للفصل
                    xticklabels=labels, yticklabels=labels,
                    annot_kws={"size": 14}, linewidths=.5, linecolor='black')
        plt.title(f'مصفوفة الارتباك - {model_name}', fontsize=18, pad=20)

```

```

plt.xlabel('التصنيف المتوقع', fontsize=14)
plt.ylabel('التصنيف الحقيقي', fontsize=14)
plt.xticks(fontsize=12, rotation=45) # تدوير العناوين السفلية إذا كانت طويلة
plt.yticks(fontsize=12, rotation=0)
plt.tight_layout()
plt.show()

# -----

print("-----")

# =====
# ورسم أهمية الميزات مع GridSearchCV تدريب وتقييم 6.
# =====
def train_evaluate_random_forest_with_gridsearch(
    X_train_selected: np.ndarray,
    X_test_selected: np.ndarray,
    y_train: pd.Series,
    y_test: pd.Series,
    feature_names: List[str], # قائمة بأسماء الميزات المستخدمة للتدريب
    feature_selection_method: str
) -> Optional[Dict[str, Any]]:
    """
    ورسم أهمية الميزات مع GridSearchCV تدريب وتقييم نموذج.

    المدخلات:
    X_train_selected (array-like): بيانات التدريب بعد اختيار الميزات.
    X_test_selected (array-like): بيانات الاختبار بعد اختيار الميزات.
    y_train (array-like): المتغير الهدف لمجموعة التدريب.
    y_test (array-like): المتغير الهدف لمجموعة الاختبار.
    feature_names (List[str]): قائمة بأسماء الميزات المقابلة لـ X_train_selected.
    feature_selection_method (str): اسم طريقة اختيار الميزات المستخدمة (للتسمية).

    المخرجات:
    Optional[Dict[str, Any]]: قاموس يحتوي على نتائج التقييم وأفضل المعلمات والنموذج، أو None عند الفشل.
    """
    step_number = 4 if "Chi-Square" in feature_selection_method else 5
    print(f"\n\n=====")
    print(f"GridSearchCV مع {feature_selection_method} على ميزات Random Forest تطبيق وتقييم {step_number} الخطوة")
    print(f"=====")

    # --- التحقق من المدخلات ---
    if X_train_selected is None or X_test_selected is None or y_train is None or y_test is None:
        print(f"\nخطأ: بيانات الإدخال غير مكتملة لتدريب نموذج {feature_selection_method}.")
        return None
    if X_train_selected.shape[0] == 0 or X_test_selected.shape[0] == 0:
        print(f"\nخطأ: مجموعة بيانات التدريب أو الاختبار فارغة بعد اختيار ميزات {feature_selection_method}.")
        return None
    if len(feature_names) != X_train_selected.shape[1]:
        print(f"\nخطأ: عدد أسماء الميزات ({len(feature_names)}) لا يتطابق مع عدد الأعمدة في بيانات التدريب ({X_train_selected.shape[1]}).")
        print(f"أسماء الميزات المتوقعة: {feature_names}")
        return None

    # --- تعريف المصنف وشبكة البحث ---
    # مفيد إذا كانت الفئات غير متوازنة 'balanced' class_weight= استخدام
    rf_classifier = RandomForestClassifier(random_state=42, class_weight='balanced')
    # تعديل شبكة البحث لتشمل خيارات أكثر أو أقل حسب الحاجة وقوة الجهاز
    param_grid_rf = {
        "n_estimators": [100, 200, 300], # عدد الأشجار
        "max_depth": [None, 10, 20, 30], # أقصى عمق للشجرة (None غير محدود)
        "min_samples_split": [2, 5, 10], # أقل عدد عينات لتقسيم العقدة
        "min_samples_leaf": [1, 3, 5], # أقل عدد عينات في الورقة النهائية
        "criterion": ["gini", "entropy"], # معيار التقسيم
        "max_features": ['sqrt', 'log2', None] # عدد الميزات للنظر فيها عند البحث عن أفضل تقسيم
    }

    try:
        # --- GridSearchCV تهيئة ---
        # cv=5 5-fold cross-validation (قيمة شائعة)
        # scoring='f1_weighted' قد يكون مقياس أفضل من 'accuracy' للبيانات غير المتوازنة
        grid_search_model = GridSearchCV(
            estimator=rf_classifier,
            param_grid=param_grid_rf,
            cv=5,
            scoring="f1_weighted", # تغيير المقياس هنا
            n_jobs=-1, # استخدام كل الأنوية المتاحة
            verbose=1 # عرض بعض المعلومات أثناء البحث
        )

        print(f"\nRandom Forest (مقياس: f1_weighted) تحسين (Grid Search) جاري البحث الشبكي...")
        # --- تطبيق البحث الشبكي ---
        grid_search_model.fit(X_train_selected, y_train)

        # --- الحصول على أفضل نموذج ونتائج البحث ---
        best_rf_classifier_model = grid_search_model.best_estimator_
        print(f"\nأفضل المعلمات الفائقة التي تم العثور عليها بواسطة GridSearchCV ({feature_selection_method}):")
        print(grid_search_model.best_params_)
        print(f"(Best CV Score - {grid_search_model.scoring}): {grid_search_model.best_score_:.4f}")

        # --- التقييم على مجموعة الاختبار ---
        print("\nإجراء التنبؤات على مجموعة الاختبار باستخدام أفضل نموذج")
        y_pred_model = best_rf_classifier_model.predict(X_test_selected)

        # --- عرض نتائج التقييم ---
        model_full_name = f"Random Forest with {feature_selection_method} Features (Optimized)"
        evaluation_results_model = evaluate_model(model_full_name, y_test, y_pred_model)
        print_evaluation_results(evaluation_results_model) # ستطبع النتائج وترسم مصفوفة الارتباك

        # --- رسم أهمية الميزات ---
        if hasattr(best_rf_classifier_model, 'feature_importances_'):
            importances = best_rf_classifier_model.feature_importances_
            feature_importance_df = pd.DataFrame({
                'Feature': feature_names,

```

```

        'Importance': importances
    }).sort_values(by='Importance', ascending=False).reset_index(drop=True)

    print(f"\nأهمية الميزات للنموذج ({model_full_name}):")
    try:
        from IPython.display import display
        display(feature_importance_df)
    except ImportError:
        print(feature_importance_df.to_string())

    # رسم بياني لأهمية الميزات
    plt.figure(figsize=(10, max(5, len(feature_names) * 0.6)))
    barplot = sns.barplot(x='Importance', y='Feature', data=feature_importance_df,
                          palette='rocket', hue='Feature', dodge=False, legend=False)
    plt.title(f'أهمية الميزات - {model_full_name}', fontsize=18, pad=20)
    plt.xlabel('الأهمية (Gini Importance)', fontsize=14)
    plt.ylabel('الميزة', fontsize=14)
    plt.xticks(fontsize=12)
    plt.yticks(fontsize=12)
    # إضافة قيم الأهمية على الإعمدة
    for index, value in enumerate(feature_importance_df['Importance']):
        plt.text(value, index, f'{value:.3f}', va='center', ha='left', fontsize=10, color='black')
    plt.tight_layout()
    plt.show()

# -----

# --- إرجاع النتائج والنموذج ---
results = {
    "evaluation": evaluation_results_model,
    "best_params": grid_search_model.best_params_,
    "best_cv_score": grid_search_model.best_score_,
    "model": best_rf_classifier_model # إرجاع النموذج المدرب
}
return results

except NotFittedError:
    print(f"\nخطأ NotFittedError: محاولة استخدام GridSearchCV قبل تدريبه (fit) على ميزات {feature_selection_method}.")
    import traceback
    traceback.print_exc()
    return None

except ValueError as e:
    print(f"\nحدث خطأ ValueError أثناء تدريب أو تقييم النموذج ({feature_selection_method}): {e}")
    print("قد يكون السبب مشاكل في البيانات المدخلة أو المسميات الفائقة.")
    import traceback
    traceback.print_exc()
    return None

except Exception as e:
    print(f"\nحدث خطأ غير متوقع أثناء تطبيق وتقييم نموذج Random Forest على ميزات {feature_selection_method}: {e}")
    import traceback
    traceback.print_exc()
    return None

# =====
# 7. تنفيذ السكريبت الرئيسي
# =====
def main():
    """
    الدالة الرئيسية لتنفيذ خطوات المشروع
    """
    print("*****")
    print("* بدء تنفيذ سكريبت تطبيق نماذج التعلم الآلي مع اختيار الميزات *")
    print("*****")

    # --- الإعدادات الرئيسية ---
    # تأكد من أن هذا الملف موجود في نفس المجلد أو حدد المسار الكامل
    data_file_path = "kidney_disease_clean_final.csv"
    target_column = "classification" # اسم العمود الهدف
    test_set_size = 0.20 # نسبة بيانات الاختبار (20%)
    random_seed = 42 # لضمان تكرار النتائج
    n_features_to_select = 8 # (يمكن تعديله)

    print(f"\nالإعدادات:")
    print(f"  - ملف البيانات: {data_file_path}")
    print(f"  - العمود الهدف: {target_column}")
    print(f"  - حجم مجموعة الاختبار: {test_set_size*100:.0f}%")
    print(f"  - البذرة العشوائية: {random_seed}")
    print(f"  - عدد الميزات للاختيار: {n_features_to_select}")
    # -----

    # --- الخطوة 1: تحميل وتقسيم البيانات ---
    data_load_result = load_and_split_data(
        data_file_path, target_column, test_set_size, random_seed
    )

    # التحقق من نجاح تحميل البيانات قبل المتابعة
    if data_load_result is None:
        print("\nفشل تحميل البيانات أو تقسيمها. لا يمكن المتابعة. إنهاء التنفيذ.")
        return # الخروج من الدالة الرئيسية

    X_train, X_test, y_train, y_test = data_load_result
    # -----

    # --- Chi-Square الخطوة 2: اختيار الميزات باستخدام ---
    # الأصلية عن طريق الخطأ داخل الدوال DataFrames لتجنب تعديل (.copy()) استخدام نسخ
    chi2_selection_result = select_features_chi2(
        X_train.copy(), y_train.copy(), X_test.copy(), n_features_to_select
    )

    # تغريغ النتائج في متغيرات منفصلة
    X_train_selected_chi2, X_test_selected_chi2, chi2_scores_df_top, selected_features_chi2_names = (None, None, None, None)
    if chi2_selection_result is not None:
        X_train_selected_chi2, X_test_selected_chi2, chi2_scores_df_top, selected_features_chi2_names = chi2_selection_result

```

```

# -----

# --- RFE الخطوة 3: اختيار الميزات باستخدام ---
rfe_selection_result = select_features_rfe(
    X_train.copy(), y_train.copy(), X_test.copy(), n_features_to_select
)

# تفرغ النتائج في متغيرات منفصلة
X_train_selected_rfe, X_test_selected_rfe, selected_features_rfe_names = (None, None, None)
if rfe_selection_result is not None:
    X_train_selected_rfe, X_test_selected_rfe, selected_features_rfe_names = rfe_selection_result
# -----

# --- تخزين النتائج النهائية للنماذج ---
final_results = {}

# --- Chi-Square على ميزات Random Forest الخطوة 4: تدريب وتقييم ---
if X_train_selected_chi2 is not None and X_test_selected_chi2 is not None and selected_features_chi2_names is not None:
    results_rf_chi2 = train_evaluate_random_forest_with_gridsearch(
        X_train_selected_chi2, X_test_selected_chi2, y_train, y_test,
        selected_features_chi2_names, # تمرير أسماء الميزات المختارة
        "Chi-Square"
    )
    if results_rf_chi2:
        final_results["Random Forest (Chi-Square)"] = results_rf_chi2
else:
    print("\n\n=====")
    print(" .بنجاح في الخطوة 2 Chi-Square تخطي الخطوة 4: لم يتم إعداد بيانات ميزات ")
    print("=====")
# -----

# --- RFE على ميزات Random Forest الخطوة 5: تدريب وتقييم ---
if X_train_selected_rfe is not None and X_test_selected_rfe is not None and selected_features_rfe_names is not None:
    results_rf_rfe = train_evaluate_random_forest_with_gridsearch(
        X_train_selected_rfe, X_test_selected_rfe, y_train, y_test,
        selected_features_rfe_names, # تمرير أسماء الميزات المختارة
        "RFE"
    )
    if results_rf_rfe:
        final_results["Random Forest (RFE)"] = results_rf_rfe
else:
    print("\n\n=====")
    print(" .بنجاح في الخطوة 3 RFE تخطي الخطوة 5: لم يتم إعداد بيانات ميزات ")
    print("=====")
# -----

# --- ملخص النتائج النهائية والمقارنة ---
print("\n\n=====")
print("ملخص النتائج النهائية والمقارنة")
print("=====")

summary_data = []
for model_name, results in final_results.items():
    if results and results.get("evaluation"):
        eval_metrics = results["evaluation"]
        summary_data.append({
            "Model": model_name,
            "Feature Selection": "Chi-Square" if "Chi-Square" in model_name else "RFE",
            "Accuracy": eval_metrics.get('Accuracy'),
            "F1 Score (Weighted)": eval_metrics.get('F1 Score (Weighted)'),
            "Precision (Weighted)": eval_metrics.get('Precision (Weighted)'),
            "Recall (Weighted)": eval_metrics.get('Recall (Weighted)'),
            "Best CV Score (f1_weighted)": results.get('best_cv_score'),
            "Best Params": results.get('best_params')
        })

if summary_data:
    summary_df = pd.DataFrame(summary_data)
    summary_df = summary_df.round(4) # تقريب الأرقام العشرية
    print("\nجدول مقارنة أداء النماذج النهائية:")
    try:
        from IPython.display import display
        display(summary_df)
    except ImportError:
        print(summary_df.to_string())

# --- تحديد النموذج الأفضل بناءً على مقياس معين (مثل F1 Score) ---
best_model_info = summary_df.loc[summary_df['F1 Score (Weighted)'].idxmax()]
print("\n--- النموذج الأفضل بناءً على أعلى F1 Score (Weighted) ---")
print(f"النموذج: {best_model_info['Model']}")
print(f"F1 Score: {best_model_info['F1 Score (Weighted)']:.4f}")
print(f"Accuracy: {best_model_info['Accuracy']:.4f}")
print(f"أفضل المعلمات: {best_model_info['Best Params']}")
print("-----")

else:
    print("\nلم يتم الحصول على نتائج لأي من النماذج لإنشاء ملخص.")

print("\n\n*****")
print(" .انتهى تنفيذ السكريبت بنجاح . *")
print("*****")

# =====
# نقطة الدخول لتشغيل السكريبت
# =====
if __name__ == "__main__":
    استدعاء الدالة الرئيسية لتشغيل كل الخطوات
    main()

```

محتوى بيانات أمراض الكلى المزمنة (CKD)

هذا المحتوى هو تمثيل للبيانات الموجودة في نهاية الملف المرفق، بصيغة CSV.

يمكن نسخ هذا المحتوى ولصقه في محرر نصوص وحفظه كملف CSV.

تأكد من حفظ الملف بالاسم 'kidney_disease_clean_final.csv' ليعمل سكريبت بايثون بشكل صحيح.

تأكد من أن الترميز هو UTF-8 عند الحفظ إن أمكن.

age,bp,sg.al,su,pc.pcc,ba,bgr bu se,sod,pot,hemo,pcv, we, htn,dm,cad,appet,pe.ane classification
48,80,1.02,1.0,1.0,0,121,36,1.2,137.5418006,4.628064516,15.4,32,72,1.4,1.0,0,0,0
7,50,102,4,0,1,0,0,147.9435028,18,0.8,137.5418006,4.628064516,11.3,26,56,0,3,1,0,0,0,0
62,80,1.01,2,3,1,0,0,423,53,1.8,137.5418006,4.628064516,9.6,19,70,0,4,1,1,0,1,0
5.11.2.20.62.1.3.1.1.1.0 48,70,1.005,4,0,0,1,0,117.56,3.8,111.2
51.80.1.01.2.0,1,0,0,106,26,1.4,137.5418006,4.628064516,11.6,23,68,0.3,1.0,0,0,0
60,90,1.015,3,0,2,0,0,74,25,1.1,142,3,2,12,2,27,72,1,4,1,0,1,0,0
68,70,1.01,0,0,1,0,0,100,54,24,104,4,12,4,24,92,0,3,1,0,0,0,0
1.137.5418006,4.628064516,124,32,64,0,4,1,0,1,0,0
24.76.502
2,100,1.015,3.0,0S
8.60.1.9.137.5418006,4.628064516,10.8,21.88,1.4,1.0,0,1.0
7,72,114,3,79.5,17,18,1,4,1,1,0,1,0 .53,90,1.02,2,0,0,1,0,70
50,60,1.01,2,4,0,1,0,490,55,4,137.5418006,4.628064516,9.4,16,92,1,4,1,0,0,1,0
63.70,1.01,3,0,0,1,0,380,60,2,7,131,4,2,10,8,20,43,1,4,1,1,0,0
68.70,1.015,3,1,1,1,0,0,208,72,2,1,138,5,8,9,7,16,19,1,4,2,1,1,0,0
68,70,1.01742
6,135,3,4,9,8,44,92,1.4,2.1.1.0.0 1428571.2.0.0,98,86,4
68,80,1.01,3,2
1,130,6,4,5,6,4,10,1,4,2,1,1,0,0
40,80,1.015,3,0,1,0,0,76,162,9,6,141,4,9,7,6,12,39,1,3,1,0,0,1.0
2,138,4,1.12,6,44,92,0,3,1,0,0,0,0 47,70,1.015,2,0,1,0,0,99,46,2
47.80.1.017428977,1.014164306,0.451428571,2,0,0,114,87,5.2,139,3,7,12,1.44,92,1.3,1.1,0,0,0
3,135,4,3,12,7,25,13,1,4,2,0,0,0,0 60,100,1.025,0,3,1,0,0,263,27,1
10,3,18,49,1,3,2,0,0,0,0,1.015,1,0,0,1,0,100,31,1.6,137.5418006,4.628064516 ,62,60
5.2,7,7,12,84,1,4,2,1,1,1,0
61,80,1.015,2,0,0,0
14164306,0.451428571,2,0,0,147.9435028,180,76,4,5,4.628064516,10,9,20,57,1,4,2,0,0,0,0
60,90,1.01742897
,48,80,1.025,4,0,0
163,7,7,136,3,8,9,8,20,64,1,3,1,0,0,1.0
21,70,1.01,0,0,1,0,0
9435028,57,5,1240106,3.07519685,137.5418006,4.628064516,12.53458213,44,92,0,3,1,1,0,1,0
42,100,1.015,4,0,0
17,9435028,50,1,4,129,4,11,1,27,77,1,3,1,1,0,0,0
61,60,1.025,0,0,1,0,0,108,75,1,9,141,5,2,9,9,17,78,1,4,1,0,0,1,0
75,80,1.015,0,0
2,4,140,3,4,11,6,23,4,1,4,1,1,0,0,0
87,27,130,4,125,25,88,1,4,2,0,1,0,0
69,70,1.01
75,70,1.017428977,1,3,2,0,0,123,31,1,4,137.5418006,4.628064516,12.53458213,44,92,0,4,1,0,0,0,0
68,70,1.005,1,0,0,1,0,147.9435028,28,14,137.5418006,4.628064516,12,9,26,92,0,3,2,0,0,0,0
53458213,44,92,1,2,1,0,0,0,0 51.39588689,70,1.017428977,1.014164306,0.451428571,2,0,0,93,155,7,3,132,4,9,12
73,90,1.015,3,0,0,1,0,107,33,1,5,141,4,6,10,1,18,72,0,3,1,1,0,0,0
61,90,1.01,1,1,1,0,0,159,39,1,5,133,4,9,11,3,22,88,1,4,1,1,0,0,0
60,100,1.02
55,2,5,137.5418006,4.628064516,10,1,17,92,1,3,1,1,0,0,0
70,70,1.01,10,1,02
153,5,2,137.5418006,4.628064516,12.53458213,44,92,0,4,1,1,0,0,0
65,90,1.02,2,1,1
5418006,4.628064516,12,24,90,1,4,1,1,0,1,0 39,2,137
76,70,1.015,1,0,1,0,0,92,29,1,8,133,3,9,10,3,20,92,1,3,1,0,0,0,0
69,80,1.02,3,0,1,0,0,147.9435028,103,4,1,132,5,9,12,5,44,92,1,3,1,0,0,0,0
82,80,1.01,2,2,2,0,0,140,70,3,4,136,4,2,13,28,90,1,4,1,0,0,0,0
46,90,1.01,2,0,0,0,0,99,80,2,1,137.5418006,4.628064516,11,1,20,83,1,3,0,0,0,0,0
7.9435028,20,0,7,137.5418006,4.628064516,12.53458213,44,92,0,3,1,0,1,0,0
45,70,1
47,100,1.01,0,0,1,0,0,204,29,1,139,4,2,9,7,21,84,1,3,1,0,0,1.0
35,80,1.01,1,0,2,0,0,79,202,10,8,134,3,4,7,9,12,73,0,4,1,0,0,0,0
3,134,4,8,9,7,16,92,1,4,1,1,1,0,0 54,80,1.01,3,0,0,0,0,207,77,6
54,80,1.02,3,0,0,0,0,208,89,5,9,130,4,9,9,3,44,92,1,4,1,1,0,0
48,70,1.015,0,0,1,0,0,124,24,1,2,142,4,2,12,4,25,59,0,4,1,0,0,0,0
11,80,1.01
18,137.5418006,4.628064516,15,33,80,0,3,1,0,0,0,0
25,4,10,17,33,1,4,1,0,1,0,0
73,70,1.00
137.5418006,4.628064516,9,7,17,35,1,4,1,1,0,1.0
60,70,1.01,2,0,0,1
53,60,1.01742897
451428571,2,0,0,91.114,3,25,142,4,3,8,6,16,10,1,4,1,1,1,0,6430
54,100,1.015,3,0,1
162,66,1,6,136,4,4,10,3,21,92,1,4,1,1,1,0,0
53,90,1.015,0,1
10,9,22,42,0,3,1,1,0,1.0 ,19435028,38,2,2,137,5418006,4.628064516
62,80,1.015,0
137.5418006,4.628064516,13,6,28,79,1,4,1,0,0,0,0
63,80,1,0
2,13,28,90,1,3,2,0,0,0,0 17.51240106,34,136,4
35,80,1.005,3,0,1,0,0,147.9435028,57,5,1240106,3.07519685,137.5418006,4.628064516,9,5,16,92,0,3,1,0,1,0,0
10,2,18,12,1,4,2,1,1,0,0 ,76,70,1.015,3,4,0,1,0,147.9435028,164,9,7,131,4,4
76,90,1.017428977,1.014164306,0.451428571,1,0,0,93,155,7,3,132,4,9,12.53458213,44,92,1,4,2,1,0,0,0
73,80,1.02,2,0,0,0,0,253,142,4,6,138,5,8,10,5,21,67,1,4,2,0,0,0,0
59,100,1.017428977,1.014164306,0.451428571,2,0,0,147.9435028,96,6,4,137.5418006,4.628064516,6,6,44,92,1,4,1,0,0,1,0
67,90,1.02,1,0,0,1
53458213,44,92,1,3,1,0,0,0,0
67,80,1.01
8,12,53458213,44,92,0,3,1,0,1,0,0
15,60,1.02,3,0,1,0,0
38,4,11,21,71,1,4,1,0,0,0,0
1,6,1,131,3,7,7,5,15,92,0,3,1,0,0,1,0
46,70,1.015,1,0,1
55,80,1.01,0,0,1,0,0,146,57,5,1240106,3.07519685,137.5418006,4.628064516,9,8,44,92,0,3,0,0,0,0,0
44,90,101,1,0,1,0,0,147,9435028,20,1,1,137.5418006,4.628064516,15,36,92,0,0,1,0,0,0,0
67,70,1.02,2,0,1,0,0,150,55,1,6,131,4,8,12,53458213,1,92,1,4,1,0,1,0,0
3,07519685,137.5418006,4.628064516,12,53458213,44,92,0,3,1,1,0,0,0
45,80
65,70,1.01,2,0,1,1,0,112,73,3,3,137.5418006,4.628064516,10,9,25,92,0,3,1,0,0,0,0
26,70,1.015,0,4,1,0,0,250,20,1,1,137.5418006,4.628064516,15,6,40,64,0,4,1,0,0,0,0
61,80,1.015,0,4,1,0,0,360,19,0,7,137,4,4,15,2,32,77,1,4,1,0,0,0,0
46,60,1.01,1,0,1,0,0,163,92,3,3,141,4,9,8,16,27,1,4,1,0,0,0,0
3,137.5418006,4.628064516,10,3,44,92,1,4,1,0,1,0,0 64,90,1.01,3,3,0,1,0,147,9435028,35,1
51.39588689,1
29,107,6,7,132,4,4,4,8,2,58,1,3,1,0,1,1,0
31,4,8,9,1,17,59,1,3,1,0,0,0,0
56,90,1.015,2
5,76,50259067,1.015,1,0,1,0,0,147.9435028,16,0,7,138,3,2,8,1,44,92,0,3,1,0,0,1,0
48,80,1.005,4,0,0,0,1,133,139,8,5,132,5,5,10,3,24,0,4,1,0,1,0,0

67,70,1.01,1.0,1.0,0,102,48,3.2,137,5.11.9.22.66,1.4,1.0,1.0,0
70,80,1.017428977,1.014164306,0.451428571,2.0,0,158,85,3.2,141,3.5,10.1,18,92,1.3,1.0,1.0,0
56,80,1.01,1.0,1.0,0,165,55,1.8,137,5418006,4.628064516,13.5.28.15.1.4.1.1.1.0.0
.74,80,1.01
0,86,1.4.1.0.0.0.0
45,90,1.017428977,1.014164306,0.451428571,2.0,0,360,45,2.4,128,4.4.8.3,17,51,1.4,1.0,0,0,0
53458213,44,92,1.3,1,1,1,0,0 38,70,1.017428977,1.014164306,0.451428571,2.0,0,104,77,1.9,140,3.9.12
48,70,1.015,1.0,1.0,0,127,19,1.134.3.6.12.53458213,44,92,1.4.1.0.0.0.0
59,70,1.01,3.0,0,0,0,76,186,15,135,7.6.7.1.10.39,1.3.1.1.1.1.0
70,70,1.015,2.0,451428571,2.0,0,147,9435028,46,1.5,137,5418006,4.628064516,9.9,44,92,0,4,1,1,1,0,0
56,80,1.017428977,1.014164306,0.451428571,2.0,0,415,37,1.9,137,5418006,4.628064516,12.53458213,44,92,0,4,1.0,0,0,0
70,100,1.005,1.0,0,1.0,169,47,2.9,137,5418006,4.628064516,11.1.20,54,1.4,1,1,0,0,0
58,110,1.01,4.0,1.0,0.251.52,2.2.137,5418006,4.628064516,12.53458213,44,25,1.1.1.0.0.0.0
50,70,1.02,0,0,1,0,0,109,32,1.4,139,4.7,12.53458213,44,92,0,3,1.1.0.0.0
63,100,1.01,2.2.1.0,1.280,35,3.2,143,3.5,13,28,90,1.3,2,0,0,0,0
56,70,1.015,4,1,1,0,0,210,26,1.7,136,3.8,16,1.40,22,0,3,1,0,0,0,0
10,4,21,52,1,4,2,0,0,0,0,71,70,1.01,3,0,0,1,1,219,82,3.6,133,4.4
73,100,1.01,3,2,0,1,0,295,90,5,6,140,2.9,9,2,18,65,1,4,2,1,0,0,0
137,4.5,11.6,24,16,0,4,1,0,0,0,0 65,70,1.01,0,0,1,0,0,93,66
1,3,1,0,1,1,0 53458213,44,92 62,90,1.015,1,0,1,0,0,94,25,1.1,131,3.7,12
60,80,1.01,1,1,1,0,0.172,32,2.7,137,5418006,4.628064516,11.2,24,92,0,4,2,1,0,0,0
2,132,3.8,10,20,83,1,4,1,1,1,0,0 65,60,1.015,1,0,1,0,0,91,51,2
50,140,1.017428977,1.014164306,0.451428571,2.0,0,101,106,6.5,135,4.3,6,2,6,54,1.4,1.1.0,1.0
2,20,5,1,4,1,1,1,0,0 56,180,1.017428977,0,4,0,0,0,298,24,1.2,139,3,9,11
34,70,1.015,4,0,0,0,0,153,22,0,9,133,3.8,12,53458213,44,92,0,3,1,0,1,0,0
4,139,5,7,11,3,21,7,0,3,1,0,0,0,0 88,80,4,71,90,1.015,2,0,0
17,60,1.01,0,0,1,0,0,92,32,2.1,141,4,2,13,9,40,65,0,3,1,0,0,0,0
76,70,1.015,2,0,0,1,0,226,217,10,2,137,5418006,4.628064516,10,2,24,23,1,3,1,1,1,1,0
55,90,1.017428977,1.014164306,0.451428571,2,0,0,143,88,2,137,5418006,4.628064516,12,53458213,44,92,1.4,1.1.1.0.0
65,80,1.015,0,0,1,0,0,115,32,11.5,139,4,14,1,30,63,0,3,1,0,0,0,0
50,90,1.017428977,1.014164
8571,2,0,0,89,118,6,1,127,4,4,6,5,60,1,4,1,0,1,1,0
55,100,1.01
45,80,1,011
54,70,1.01742
63,90,1.015,0,0,1
1,4,1,0,0,0,0
25,3,0,3,1,0,0,0,0
1428571,2,0,0,233,50,1.1,9,137,5418006,4.628064516,11,7,44,92,0,4,1,0,0,0,0
142,3,8,11,7,22,13,0,3,1,0,0,0,0
4,128,5,4,10,20,82,1,4,2,0,0,0,0 65,80,1.01,3,3,1,0,0,294
51,39588689,6
7,9435028,34,1.2,137,5418006,4.628064516,10,8,21,92,0,3,1,0,0,0,0
.61,90,1
12,60,1,015
3,07519685,137,5418006,4.628064516,12,53458213,44,90,0,4,1,1,0,1,0
51,1.8,137,5418006,4.628064516,12,1,44,4,0,3,1,0,0,0,0
47,80,1.01,0,0,0,0,0
7,9435028,28,0,9,137,5418006,4.628064516,124,32,52,0,3,1,0,0,1,0
51,39588689,70,1.015,4,0,1,0,0,104,16,0,5,137,5418006,4.628064516,12,53458213,44,92,0,3,1,0,1,0,0
51,39588689,70,1.02,0,0,2,0,0,219,36,1,3,139,3,7,12,5,25,90,0,3,1,0,0,0,0
55,70,1.01,3,0,1,0,0,99,25,1,2,137,5418006,4.628064516,11,4,44,92,0,3,1,1,1,0,0
60,70,1.01,0,0,1,0,0,140,27,1,2,137,5418006,4.628064516,12,53458213,44,92,0,3,1,0,0,0,0
72,90,1.017428977,1.014164306,0.451428571,2,0,0,308,36,2,5,131,4,3,12
70,90,1.015,0,0,1,0,0,144,125,4,136,4,6,12,25,76,1,4,1,1,0,1,0
.71,60,1.015,4,0,1
8,125,5,3,136,4,9,11,4,23,29,1,4,1,1,0,1,0
52,90,1.015,4,3,0,0,0,224,166,5,6,133,47,8,1,11,46,1,4,1,0,0,1,0
75,70,1.025,1,0,1,0,0,158,49,1,4,135,4,7,11,1,44,92,1,3,1,1,1,0,0
50,90,1.01,2,0,0,1,1,128,208,9,2,134,4,8,8,2,10,31,0,3,1,1,1.1.0
53458213,44,92,1.4,1,1,0,0,0
5,50,1.01,0,0,1,0,0,147,9435028,25,0,6,137,5418006,4.628064516,11,8,24,21,0,3,1,0,0,0,0
50,76,50259067,1.017428977,1.014164306,0.451428571,2,0,0,219,176,13,8,136,4,5,8,6,12,25,1,3,1,0,1,1,0
70,100,1.015,4
47,100,1.01
5,3,136,4,9,12,25,1,1,3,1,0,0,0,0
1,451428571,2,0,0,122,57,51240106,16,9,138,5,2,10,8,21,3,0,4,1,0,0,0,0
48,80,1.015,0,2,1,0,0,214,24,13,140,4,13,2,27,92,0,4,1,1,0,0,0
46,90,1.02,1.014164306,0.451428571,1,0,0,213,68,2,8,146,6,3,9,3,44,92,1,4,1,0,0,0,0
45,60,1.01,2,0,0,1,0,268,86,4,134,5,1,10,17,84,1,4,1,0,0,0,0
73,76,50259067,1.01,1,0,2,0,0,95,51,1,6,142,3,5,12,53458213,44,92,0,0,1,0,0,0,0
41,70,1.015,2,0,0,0,1,147,9435028,68,2,8,132,4,1,11,1,21,92,1,3,1,0,1,1,0
.69,70,1.01,0,4
56,40,12,142,5,6,12,53458213,44,92,0,3,1,0,0,0,0
67,70,1.01,1,0,1,0,0,147,9435028,106,6,137,4,9,6,1,7,60,1,3,1,0,0,1,0
53458213,44,92,0,4,1,0,0,0,0 3,12 72,90,1.017428977,1.014164306,0.451428571,2,0,0,84,145,7,1,135,5
53458213,44,92,0,4,1.0,0,0,0 41,80,1.015,1,4,1,0,0,210,165,18,135,4,7,12
60,90,1.01,2,0,1,0,0,105,53,2,3,136,5,2,11,1,21,6,0,3,1,0,0,0,0
57,90,1.015,5,0,0,0,1,147,9435028,322,13,126,4,8,8,12,41,1,4,2,1,1,1,0
53,100,1.01,1
60,60,1.01,3,1
1,139,4,12,53458213,44,92,0,4,1,0,0,0,0
13,79,13,29,13,1,1,0,1,0
69,60,1.017428977,1.014164306,0.451428571,2,0,0,171,26,48,1.137,5418006,4.628064516,12,53458213,44,92,1,3,1,1,0,0,0
65,70,1.02,1,0,0,0,0,139,29,1.137,5418006,4.628064516,10,5,20,92,1,3,1,0,1,0,0
8,60,1.025,3,0,1,0,0,78,27,0,9,137,5418006,4.628064516,12,3,29,62,0,3,1,1,1,0,0
76,90,1.017428977,1.014164306,0.451428571,2,0,0,172,46,1,7,141,5,5,9,6,18,92,1,4,1,0,0,1,0
39,70,1.01,0,0,1,0,0,121,20,0,8,133,3,5,10,9,20,92,0,4,1,0,0,0,0
55,90,1.01
4,2,132,3,4,8,3,10,27,1,4,1,1,1,1,0
56,90,1.005,4,3,0,0,0,242,132,16,4,140,4,2,8,4,14,92,1,4,1,1,1,1,0
50,70,1.02,3,0,1,1,1,123,40,1,8,137,5418006,4.628064516,11,1,24,44,0,3,1,0,0,0,0
66,90,1.015,2,0,1,0,1,153,76,33,137,5418006,4.628064516,12,53458213,44,92,0,3,1,1,0,0,0
62,70,1.025,3,0,0,0,0,122,42,1,7,136,4,7,12,6,27,73,1,4,1,0,0,0,0
71,60,1.02,3,2,1,1,0,424,48,1,5,132,4,10,9,19,92,1,4,2,0,0,0,0
.59,80,1.01
81,60,1.0174
1,5,10,4,23,9,0,4,1,1,0,0,0
128571,2,0,0,148,39,2,1,147,4,2,10,9,23,86,1,4,2,1,1,0,0
7,9435028,57,51240106,3,07519685,137,5418006,4.628064516,14,3,30,3,1,4,1,0,0,0,0 62,76,50259067,1.015,3,0,2,0
1428571,2,0,0,204,34,1,5,124,4,1,9,8,25,56,0,4,1,0,0,0,0 59,70,1.017428977,1.014164306,0
46,80,1.01,0,0,1,0,0,160,40,2,140,4,1,9,15,75,1,3,1,1,0,1,0
14,76,50259067,1
14,3,28,87,0,4,1,1,1,0,0,5,0,8,137,4,2
27,60,1
8571,2,0,0,76,44,3,9,127,4,3,12,53458213,44,92,0,3,1,1,1,1,0
.60,80,1.02,0,2,2,0,0,147,94

157.51240106,3.07519685,137.5418006,4.628064516,12.53458213,44.92,0,4,1,0,0,0,0
34.70,1.02,0,0,1,0,0,139,19,0.9.137.5418006,4.628064516,12.7.30,36,0,3,1,1,0,0,0
65.70,1.015,4,4,1,1,0,307,28,1.5,137.5418006,4.628064516,11.27,62,1,4,1,0,0,0,0
51.39588689,70,1.01,0,2,1,0,0,220,68,2.8,137.5418006,4.628064516,8.7,15,92,1,4,1,0,0,1,0
5,21,88,1,4,1,0,0,0,0 66,70,1.015,2.5,1,0,0,447,41,1.7,131,3,9,12
83,70,1.02,3,0,1,0,0,102,60,2,6,115,5,7,8,7,14,24,1,3,1,1,0,1,0
1164306.0.451428571,2,0,0,111,146,7.5,141,4,7,11,23,80,0,3,1,0,0,0,0
24,0,3,1,0,0,0,0
8,13,1,29,11,0,3,1,0,0,0,0
1,7,125,3,5,8,3,11,21,0,3,1,0,0,1,0
53458213,44,41,1,3,1,0,0,0,0 2,113,3,12 60,50,1,01,0,0,1,0,0,261,58,2
62,80,1,01,1,2
17,70,1,015,1
54.70,1.017428977,1
21,90,1,01,4,0,0,1,1,1
65,80,1.015,2,1,1
42,90,1.02,2,0,0,1
72,90,1.01
13,2,29,92,0,4,1,0,0,0,0 .5,137.5418006,4.628064516
2,7,139,4,3,9,8,22,90,0,3,1,1,1,1,0
4.628064516,11.9,27,92,0,3,1,0,0,0,0
73.90,1.01,1,4,0,1,0,234,56,1.9,137.5418006,4.628064516,10,3,16,92,0,4,1,0,0,0,0
45,70,1.025,2,0,0,1,0,117,52,2,2,136,3,8,10,18,34,0,3,1,0,0,0,0
61,80,1.02,0,0,1,0,0,131,23,0,8,140,4,1,11,3,23,92,0,3,1,0,0,0,0
30,70,1.015,0,0,1,0,0,101,106,6,5,135,4,3,12,53458213,44,92,0,3,1,1,0,0,0
54,60,1.015,3,2,0,0,0,352,137,3,3,133,4,5,11,3,19,54,1,4,2,1,1,0,0
4,76,50259067
8,50,1,02,4,0,1,0,0
1,6,138,4,4,12,22,2,0,3,1,0,0,0,0
1,135,3,8,12,53458213,44,92,0,3,1,0,1,0,0
3,76,50259067,1.01,2,0,1,0,0,147,9435028,22,0,7,137.5418006,4.628064516,10,7,22,20,0,3,1,0,0,0,0
8,76,50259067,1.017428977,1.014164306,0.451428571,2,0,0,80,66,2,5,142,3,6,12,2,26,92,0,0,1,0,0,0,0
64,60,1.01,4,1,0,0,1,239,58,4,3,137,5,4,9,5,17,70,1,4,1,1,1,0,0
6,60,1.01,4,0,0,0,1,94,67,1,135,4,9,9,9,18,32,0,3,1,1,0,0,0
51.39588689,70,1.01,3,0,1,0,0,110,115,6,134,2,7,9,1,14,84,1,4,1,1,0,0,0
46,110
53458213,44,92,0,3,1,0,0,0,0 0.9,137.5418006,4.628064516,12
32,90,1.025,1,0,0,0,0,147,9435028,22,3,18,1,113,6,5,5,3,37,1,4,1,1,1,1,0
80,70,1.01,2,0,451428571,0,0,0,147,9435028,49,1,2,137.5418006,4.628064516,12,53458213,44,92,1,1,1,0,0,0,0
70,90,1.02,2,1,0,0,1,184,98,6,3,3,138,3,9,5,8,44,92,1,4,2,1,0,0,0
49,100,1.01,3,0,0,0,0,129,158,11,8,122,3,2,8,1,12,88,1,4,1,1,1,1,0
57,80,1.017428977,1.014164306,0.451428571,2,0,0,147,9435028,111,9,3,124,5,3,6,8,44,42,1,4,1,0,0,1,0
2,18,38,1,4,1,1,1,0,0 59,100,1.02,4,2,1,0,0,252,40,3,2,137,4,7,11
65,80,1.015,0,0,1,0,0,92,37,1,5,140,5,2,8,8,13,7,1,3,2,0,1,0,0
90,90,1.025,1,0,1,0,0,139,89,3,140,4,1,12,25,73,1,4,1,0,0,0,0
64,70,1.017428977,1.014164306,0.451428571,2,0,0,113,94,7,3,137,4,3,7,9,9,92,1,4,2,0,1,1,0
78,60,1.017428977,1.014164306,0.451428571,2,0,0,114,74,2,9,135,59,8,12,92,0,4,1,0,0,1,0
5,44,92,1,4,1,0,0,1,0 51.39588689,90,1.017428977,1.014164306,0.451428571,2,0,0,207,80,6,8,142,5,5,8
65,90,1.01,4,2,1,0,0,172,82,13,5,145,6,3,8,8,19,92,1,4,1,0,1,1,0
61,70,1.017428977,1.014164306,0.451428571,2,0,0,100,28,2,1,137.5418006,4.628064516,12,6,31,92,1,4,1,0,0,0,0
60,70,1.01,1,0,1,0,0,109,96,3,9,135,4,13,8,29,92,1,3,1,0,0,0,0
2,137.5418006,4.628064516,12,29,5,1,4,1,0,0,0,0 50,70,1,01,0,0,1,0,0,230,50,2
3,29,64,1,4,1,0,0,1,0 67,80,1.017428977,1.014164306,0.451428571,2,0,0,341,37,1,5,137.5418006,4.628064516,12
19,70,1.02,0,0,1,0,0,147,9435028,57,51240106,3.07519685,137.5418006,4.628064516,11,5,44,64,0,3,1,0,0,0,0
59,100,1.015,4,2,1,0,0,255,132,12,8,135,5,7,7,3,8,90,1,4,2,0,0,1,0
54,120,1.015,0,0,1,0,0,103,18,1,2,137.5418006,4.628064516,12,53458213,44,92,0,3,1,0,0,0,0
10,9,19,81,1,4,1,1,1,0,0,40,70,1.015,3,4,1,0,0,253,150,11,9,132,5,6
55,80,1.01
7,4,9,10,9,22,69,1,4,1,0,1,0,0
68,80,1.015,1
628064516,13,7,0,45,0,4,1,0,0,0,0
2,76,50259067,1
57.51240106,3.07519685,137.5418006,4.628064516,12.53458213,44,92,0,3,1,0,1,0,0 7,9435028
64,70,1.01,0,0,1
15,3,07519685,137.5418006,4.628064516,12,8,26,92,0,3,1,0,0,0,0
,
6,4,628064516,9,8,16,74,1,4,1,1,0,0,0
36,80,1,0
63,100,1.01,1,0,1,0,0
1,8,141,4,4,12,2,24,6,0,4,1,0,0,0,0
33,90,1.015,0
1,8,137.5418006,4.628064516,11,8,22,65,0,3,1,0,0,0,0
68,90,1,01,0,0
51240106,3.07519685,137.5418006,4.628064516,11,9,24,81,0,3,1,0,0,0,0
53458213,44,92,1,4,1,0,0,0,0 3,12,66,70,1.02,1,0,2,0,0,248,30,1,7,138,5
74,60,1.017428977,1.014164306,0.451428571,2,0,0,108,68,1,8,137.5418006,4.628064516,12,53458213,44,92,1,4,1,0,0,0,0
71,90,1.01,0,3,1,0,0,303,30,1,3,136,4,1,13,26,84,1,4,1,0,0,0,0
34,60,1.02,0,0,1,0,0,117,28,2,2,138,3,8,12,53458213,44,92,0,3,1,0,1,0,0
60,90,1,01,3,5,1,0,1,490,95,2,7,131,3,8,11,5,23,17,1,4,1,0,0,0,0
64,100
1,4,1,0,1,0,0
120,48,1,6,137.5418006,4.628064516,11,3,24,67,1,4,1,0,0,0,0 ,57,80,1.015,0,0,1
53458213,44,92,1,3,1,0,0,0,0 164306,0.451428571,2,0,0,124,52,2,5,137.5418006,4.628064516,12
60,70,1,01,742
191,12,114,2,9,9,6,19,30,0,4,1,0,1,0,0 59,50,1.01,3,0,0,0
60,90,1.017428977,1.014164306,0.451428571,2,0,0,269,51,2,8,138,3,7,11,5,23,92,1,4,2,0,1,0,0
50,90,1.015,1,0,0
1,9435028,57,51240106,3.07519685,137,5418006,4.628064516,12,53458213,44,92,0,3,1,0,1,0,0
51,100,1.015,2
1,6,146,4,5,12,53458213,44,92,0,3,1,1,0,0,0
37,100,1.01
1,3,137.5418006,4.628064516,15,32,40,1,3,1,0,0,0,0
53,0,3,2,0,0,1,0 137.5418006,4.628064516,7,9,14,45,70,1,01,2,0,1,0,0,113,93
65,80,1.017428977,1.014164306,0.451428571,2,0,0,74,66,2,136,5,4,9,1,13,92,1,4,2,0,1,0,0
80,70,1.015,2,2,1,0,0,141,53,2,2,137.5418006,4.628064516,12,7,28,88,1,4,1,1,1,0,0
72,100,1.017428977,1.014164306,0.451428571,2,0,0,201,241,13,4,127,4,8,9,4,16,92,1,4,1,0,0,1,0
34,90,1.015,2,0,1,0,0,104,50,1,6,137,4,1,11,9,27,92,0,3,1,0,0,0,0
628064516,11,4,24,46,1,4,1,1,1,0,0
65,70,1,015
57,70,1.015,1
45,1,5,140,3,3,10,4,19,41,0,3,1,0,0,0,0
4,96,6,3,120,3,9,9,4,16,14,1,4,2,0,1,1,0 69,70,1.01,4,3,0,1
62,90,1.02,2,1,1,0,0,169,48,2,4,138,2,9,13,4,35,10,1,3,1,0,0,0,0
64,90,1.015,3,2,0,1,0,463,64,2,8,135,4,1,12,2,28,90,1,4,1,0,0,1,0
48,100,1.017428977,1.014164306,0.451428571,2,0,0,103,79,5,3,135,6,3,6,3,7,67,1,3,2,1,0,0,0
48,110,1.015,3,0,1
15,15,2,120,5,7,8,6,14,46,1,3,2,0,0,1,0
54,90,1.025
2,12,53458213,44,92,0,3,1,1,1,1,0
59,70,1.01,1,3,0,0,0,424,55,1,7,138,4,5,12,6,25,3,1,4,2,0,0,0,0
56,90,1.01,4,1,0,1,0,176,309,13,3,124,6,5,3,1,43,50,1,4,1,1,1,1,0

40,80,1.025,0,0,1,0,0,140,10.1.2.135.5.15.36.5.0.3.1.0.0.0.1

23.80.1.025,0,0,1,0,0,70,36,1,150,4.6.17.40,90,0,3,1,0,0,0,1

45,80,1.025,0,0,1,0,0,82,49,0,6,147,4.4.15.9,34,83,0,3,1,0,0,0,1

.0,3,1.0.0.0.1

57,80,1.025

51.60,1.025.0

1.0.0.0.1

3.9,13.6,40,84,0,3,1,0,0,0,1

34,80,1.025.0.1

60,80,1.025.0.0.1

14.5.29.7.0.3.1.0.0.0.1

0.7,135,3.7,14,34,83.0.3,1,0,0,0,1 38,60,1.02.0.0,1.0.0.9

9,32,78,0,3,1,0,0,0,1

.42,80,1.02,0,0

.35,80,1.02.0

5,16.1,33,42,0,3,1,0,0,0,1

30,80,1

1.0.0.0.1

.49,80.1.02.0.0

9,3.9,17,29,52,0,3,1,0,0,0,1

.55,80,1.02.0,0,1

4,09,135,3.6.15.5.31.67,0,3,1,0,0,0.1

45,80,1.02.0.0.1.0.0.1

46

137,5,16.2.33,80,0,3,1,0,0,0.1

42,80,1.02,0,0,1,0,0,1

24,0.7,140,4.1,14.4,38,46,0,3,1,0,0,0,1

50,80,1.02,0,0

50,4.5,14.2,36,6,0,3,1,0,0,0,1

1,3,1.0,0,0,1

.55,80,1

9,13.9,36,87,0,3,1,0,0,0,1

48,80,1

.51.39588689

164306,0.451428571,2.0,0,100,49,1,140,5,163,41,79,0,3,1,0,0,0,1

.25,80,1.025,0

9,15,36,64,0,3,1,0,0,0,1

29,67,0.3,1.0.0.0.1

.23.80.1.025.0

30,80,1.021

56,80,1.025

8,30,82,0,3,1,0,0,0,1

10,52,0,3,1,0,0,0,1

47,80,1

4,92.0,3,1,0,0,0,1 582

19,80,1.02.0,0,1,0,0,107,23,0.7,141,4.2.14.4,32,92,0,3,1,0,0,0,1

52,80,1.02.0,0,1,0,0.125.22.1.2,139,4.6,16.5.31,44,0,3,1,0,0,0,1

20,60,1.025,0,0,1,0,0,147.9435028,57.51240106,3.07519685,137,4.7,14,29,43,0,3,1,0,0,0.1

46.60.1.025,0,0,1,0,0,123,46,1,135,5,15.7,38,58,0,3,1,0,0,0,1

48,60,1.02,0,0,1,0,0,112,44,1.2,142,4.9,14.5.32,86,0,3,1,0,0,0.1

24,70,1.025,0,0,1,0,0

3,36,54,0,3,1,0,0,0,1

47,80,1.01742

128571,2,0,0,93,33,0.9,144,4.5,13.3,40,75,0,3,1,0,0,0,1

55,80,1.025,0,0,1,0,0,130,50,1.2,147,5,15.5,29,83,0,3,1,0,0,0,1

20,70,1.02,0,0,1,0,0,123,44.1.135.3.8,14.6,32.51.0.3.1.0.0.0.1

60,70,1.02.0,0,1,0,0,147.9435028,57.51240106,3.07519685,137.5418006,4.628064516,16.4.31.8.0.3.1.0.0.0.1

33,80,1.025,0,0,1,0,0,100,37,1.2,142,4,16.9,40,62,0,3,1,0,0,0.1

66,70,1.02,0,0,1,0,0,94,19,0.7.135.3.9,16,29,49,0,3,1,0,0,0,1

0,0,3,1,0,0,0,1

.71,70,1.02

39,70,1.025,0,0,1,0,0,124,22,0.6,137,3,8,13.4.31.92.0.3.1.0.0.0.1

56,70,1.025,0,0,1,0,0,70,46,1.2,135,4.9,15.9,38,10,2.5.3,0,0,0,1

42,70,1.02,0,0,1,0,0,93,32,0.9.143.4.7.16.6.31,66,0,3,1,0,0,0.1

54.70.1.02.0.0.2.2.2,76,28.0.6.146.3.5.14.8,40,78,0.3.1.0.0.0.1

47,80,1.025,0,0,1,0,0,124,44,1,140,4.9,14.9,29,65,0,3,1,0,0,0,1

39,5,16.7,40,3,0,3,1,0,0,0,1

30,80,1.02.0.0.1.0

50,76.50259067

9.1.2.150.4.8.14.9,36,44,0,3,1,0,0,0,1

75.60.1.02,0,0,1,0,0,110,50,0.7,135,5,14.3,28,77,0.3.1.2.2.2.1

44,70,1.017428977,1.014164306,0.451428571,2,0,0,106,25,0.9,150,3.6,15,38,88,0,3,1,0,0,0,1

41.70.1.02.0,0,1,0,0,125,38,0.6,140,5.16.8,29,58,0,3,1,0,0,0.1

53,60,1.025,0,0,1,0,0,116,26,1,146,4.9,15.8,33,71,2.5,3,0,0,0.1

34,60,1.02,0,0,1,0,0,91,49,1.2.135.4.5,13.5,36,80,0,3,1,0,0,0,1

.73,60,1.02

5.1.40,10,0,3,1.0.0.0.1

45,60,1.02,0,0,1,2,2,114,26,0.7,141,4.2.15,31,84,0,3,1,0,0,0.1

44,60,1.025,0,0,1,0,0,96,33,09,147,4.5,16.9,29,67,0,3,1,0,0,0,1

29,70,1.02,0,0,1,0,0,127,44.1.2.145.5.14.8.36,92,0.3.1.0.0.0.1

55.70.1.02.0.0.1.0.0.107.26.1.1.137.5418006,4.628064516.17.38.62,0.3.1.0.0.0.1

33,80,1.025,0,0,1,0,0,128,38,0.6,135,3.9,13.1,33,57,0,3,1,0,0,0.1

41,80,1.02,0,0,1,0,0,122,25,0.8,138,5,17,1,29,83,0,3,1,0,0,0.1

52,80,1.02,0,0,1,0,0,128,30,12,140,4.5.15.2,40,42.0.3.1.0.0.0.1

47,60,1.02,0,0,1,0,0,137,17,0.5,150,3.5,13.6,32,73,0,3,1,0,0,0,1

6,135,4.9,13.9,36,64,0,3,1,0,0,0,1 43,80,1.025,0,0,1,0,0,81,46,0

51,60,1.02,0,0,2,0,0,129,25,12,139,5,17.2,28,75,0,3,1,0,0,0.1

46,60,1.02.0.0.1.0.0.102.27.07.142.4.9.13.2.32,10.0.3.1.0.0.0.1

56,60,1.025,0,0,1,0,0,132,18,1.1,147,4.7.13.7.33.70,0,3,1,0,0,0,1

80,70,1.02,0,0,1,0,0,147.9435028,57.51240106,3.07519685,135,4.1.15.3,36,58,0,3,1,0,0,0,1

55,80,1.02,0,0,1,0,0,104,28,0.9,142,4.8,17.3,40,76,0,3,1,0,0,0,1

39,70,1.025,0,0,1,0,0,131,46,0.6,145,5,15.6,29,86,0,3,1,0,0,0,1

44,70,1.025,0,0,1,0,0,147.9435028,57.51240106,3.07519685,137.5418006,4.628064516,13.8,36,72,0,3,1,0,0,0.1

35,76.50259067,1.02,0,0,1,2.2,99,30,0.5,135,4.9,15.4,36,46,0,3,1,0,0,0,1

58.70.1.02,0,0,1,0,0,102,48,1.2,139,4.3,15,28,75,0.3.1.0.0.0.1

61,70,1.025,0,0,1,0,0,120,29,0.7,137,3.5,17.4,40,65,0,3,1,0,0,0,1

30,60,1.02,0,0,1,0,0,138,15,1.1,135.4.4,12.53458213,44,92,0,3,1,0,0,0,1

57,60,1.02.0.0

2,5,0,3,1,0,0,0,1

65,60,1.02.0

88,0,3,1.0.0.0.1

70,60,1.01742897

64306,0.451428571,2.0,0,120,40,0.5,140,4.6,16,31,43,0,3,1,0,0,0,1

43.80,1.025,0,0,1,0,0,130,30.1.1,143,5,15.9,33,72,0,3,1,0,0,0,1

40.80.1.02.0.0,1,0,0,119,15.0.7.150.4.9.12.53458213,44,92.0.3.1.0.0.0.1

58,80,1.02,0,0,1,0,0,100,50,12,140,3.5,14,38,62,0,3,1,0,0,0,1

47,60,1.02.0

1.1,141,4.7,15.8,29,77,0,3,1,0,0,0,1

31,0,8,150,4.6,13.4,32,7,0,3,1,0,0,0,1

30,60,1.025.1
28,70,1.02,0,0,1,2,2,131,29,0,6,145,4,9,12.53458213,33,80,0,3,1,0,0,0,1
33,60,1.025,0,0,1,0,0,80,25,09,146,3,5,14.1.36,72,0,3,1,0,0,0,1
43,80,1.02,0,0,1,0,0,114,32,1,1,1,135,3,9,12.53458213,30,92,0,3,1,0,0,0,1
59,70,1.025,0,0,1,0,0,130,39,0,7,147,4,7,13,5,34,62,0,3,1,0,0,0,1
34,70,1.025,0,0,1,0,0,87,38,0,5,144,4,8,17,1,35,69,0,3,1,0,0,0,1
.23,80,1.02
17,7,34,42,0,3,1,0,0,0,1
51240106,3.07519685,136,3,5,15,4,31,52,0,3,1,0,0,0,1 24,80,1.025,0,0,1,0,0,125,57
60,60,1.02,0,0,1,0,0,134,45,0,5,139,4,8,14,2,36,7,0,3,1,0,0,0,1
25,60,1.02,0,0,1,0,0,119,27,0,5,137,5418006,4,628064516,15,2,28,84,0,3,1,0,0,0,1
44,70,1.025,0,0,1,0,0,92,40,0,9,141,4,9,14,40,70,0,3,1,0,0,0,1
62,80,1.02,0,0,1,0,0,132,34,0,8,147,3,5,17,8,32,44,0,3,1,0,0,0,1
88,42,0,5,136,3,5,13,3,36,65,0,3,1,0,0,0,1
25,70,1.02,0,0,1
32,70,1.025,0
1,142,4,5,14,3,31,62,0,3,1,0,0,0,1
63,70,1.025,0,0,1,0,0,130,37,0,9,150,5,13,4,29,68,0,3,1,0,0,0,1
44,60,1.02,0,0,1,0,0,95,46,0,5,138,4,2,15,38,71,0,3,1,0,0,0,1
37,60,1.025,0,0,1,0,0,111,35,0,8,135,4,1,16,2,38,51,0,3,1,0,0,0,1
64,60,1.02,0,0,1,0,0,106,27,0,7,150,3,3,14,4,30,75,0,3,1,0,0,0,1
22,60,1.025,0,0,1,0,0,97,18,1,2,138,4,3,13,5,30,73,0,3,1,0,0,0,1
33,60,1.017428977,1.014164306,0,451428571,1,0,0,130,41,0,9,141,4,4,15,5,40,42,0,3,1,0,0,0,1
18,25,1,144,5,17,8,31,67,0,3,1,0,0,0,1 43,60,1.025,0,0,1
38,80,1.02,0,0,1,0,0,99,19,0,5,147,3,5,13,6,32,68,0,3,1,0,0,0,1
35,70,1.025,0,0,2,0,0,82,36,1,1,150,3,5,14,5,40,86,0,3,1,0,0,0,1
65,70,1.025,0,0,2,0,0,85,20,1,142,4,8,16,1,31,88,0,3,1,0,0,0,1
29,80,1.02,0,0,1,0,0,83,49,0,9,139,3,3,17,5,28,91,0,3,1,0,0,0,1
37,60,1.02,0,0,1,0,0,109,47,1,1,141,4,9,15,36,65,0,3,1,0,0,0,1
.39,60,1.02
954,0,3,1,0,0,0,1
32,60,1.025,0,0,1,0,0,102,17,0,4,147,4,7,14,6,29,63,0,3,1,0,0,0,1
23,60,1.02,0,0,1,0,0,95,24,0,8,145,5,15,40,58,0,3,1,0,0,0,1
34,70,1.025,0,0,1,0,0,87,38,0,5,144,4,8,17,1,35,69,0,3,1,0,0,0,1
66,70,1.025,0,0,1,0,0,107,16,1,1,140,3,6,13,6,30,10,0,3,1,0,0,0,1
47,60,1.02,0,0,1,0,0,117,22,1,2,138,3,5,13,33,48,0,3,1,0,0,0,1
.74,60,1.02,0,0
5,0,3,1,0,0,0,1
35,60,1.025
1,54,0,3,1,0,0,0,1
29,80,1.02,0,0,1,0,0,70,1
3,5,13,7,42,50,0,3,1,0,0,0,1
33,80,1.025,0,0,1,0,0
1,5,15,28,4,0,3,1,0,0,0,1
87,5418006,4,628064516,17,8,32,55,0,3,1,0,0,0,1
.67,80,1.025,0,0,1,0,0,99
.73,80,1.025,0,0,1
17,3,5,14,8,33,85,0,3,1,0,0,0,1
24,80,1.02,0,0,1,0
1,145,3,5,12,53458213,44,7,0,3,1,0,0,0,1
60,80,1
1,603,1,0,0,0,1
68,60,1.025,0,0,1,0,0
5,41,1,1,139,3,8,17,4,38,62,0,3,1,0,0,0,1
30,80,1.025,0,0,1,0,0,82,42,07,146,5,14,9,33,86,0,3,1,0,0,0,1
75,70,1.02,0,0,1,0,0,107,48,0,8,144,3,5,13,6,34,4,0,3,1,0,0,0,1
69,70,1.02,0,0,1,0,0,83,42,1,2,139,3,7,16,2,38,85,0,3,1,0,0,0,1
28,60,1.025,0,0,1,0,0,79,50,0,5,145,5,17,6,39,60,0,3,1,0,0,0,1
.72,60,1.02
0,3,1,0,0,0,1
61,70,1.02
84,0,3,1,0,0,0,1
.79,80,1.025,0
3,6,16,3,28,74,0,3,1,0,0,0,1
70,80,1.02,0,0,1
5,15,1,36,89,0,3,1,0,0,0,1
1,147,3,5,16,4,41,83,0,3,1,0,0,0,1
.58,70,1.025,0,0,1
13,8,37,59,0,3,1,0,0,0,1

.64,70,1.02,0,0
71,60,1.025,0
57,51240106,0,9,140,4,8,15,2,30,71,0,3,1,0,0,0,1
62,80,1.02
5,16,1,38,50,0,3,1,0,0,0,1
.59,60,1.02,0,0,1
3,42,60,0,3,1,0,0,0,1 1,139,3,5,15
71,70,1.025,0,0,2,0,0,79,47,0,5,142,4,8,16,6,28,54,0,3,1,0,0,0,1
48,80,1.025,0,0,1,0,0,75,22,0,8,137,5,16,8,39,56,0,3,1,0,0,0,1
80,80,1.025,0,0,1,0,0,119,46,0,7,141,4,9,13,9,37,47,0,3,1,0,0,0,1
57,60,1.02,0,0,1,0,0,132,18,1,1,150,4,7,15,4,30,10,0,3,1,0,0,0,1
.63,70,1.02
5,40,74,0,3,1,0,0,0,1
16,4,31,53,0,3,1,0,0,0,1
46,70,1.02
15,80,1.025,0,0,1,0
3,9,16,7,38,57,0,3,1,0,0,0,1
51,80,1.02,0,0,1
1,2,144,3,7,15,5,34,87,0,3,1,0,0,0,1
.41,80,1.025,0,0,1,0
48,0,7,140,5,17,40,67,0,3,1,0,0,0,1
.52,80,1.025,0,0
0,8,135,3,7,15,40,58,0,3,1,0,0,0,1
36,80,1.025,0,0,1,0,0,85,16,1,1,142,4,1,15,6,32,54,0,3,1,0,0,0,1
.57,80,1.02
0,0,0,1
43,60,1.025,0,0,1,0,0,117,45,0,7,141,4,4,13,42,69,0,3,1,0,0,0,1
50,80,1.02,0,0,1,0,0,137,46,08,139,5,14,1,33,87,0,3,1,0,0,0,1
55,80,1.02,0,0,1,0,0,140,49,0,5,150,4,9,15,7,35,62,0,3,1,0,0,0,1
42,70,1.025,0,0,1,0,0,75,31,1,2,141,3,5,16,5,42,72,0,3,1,0,0,0,1
12,80,1.02,0,0,1,0,0,100,26,0,6,137,4,4,15,8,37,61,0,3,1,0,0,0,1
17,60,1.025,0,0,1,0,0,1
9,67,0,3,1,0,0,0,1
58,80,1.025,0,0,1,0,0,131,18,1,1,141,3,5,15,8,41,63,0,3,1,0,0,0,1